Computer Security

Modular Arithmetic

Sources

- *The Basics of Abstract Algebra,* Paul E. Bland, Freeman
- *Introduction to Cryptography with Java Applets*, David Bishop, Jones & Bartlett
- *Practical Cryptography*, Niels Ferguson and Bruce Schneier, Wiley
- *Concrete Mathematics*, Graham, Knuth, and Patashnik, Addison-Wesley
- *Network Security: Private Communications in a Public World*, *Second Edition* Charlie Kaufman, Radia Perlman, Mike Speciner, Prentice-Hall
- *Applied Cryptography: Protocols, Algorithms and Source Code in C, Second Edition,* Bruce Schneier, Wiley
- *Cryptography and Network Security: Principles and Practices*, William Stallings, Prentice-Hall
- and a number of web sites...

Modular Arithmetic

- ***Modular Arithmetic*** is one of the main tools provided by number theory
  - The ***quotient*** of n divided by m is $\lfloor n/m \rfloor$, where m and n are positive integers
  - The ***remainder*** of this division is called 'n mod m'
  - So, the following holds:

$$n = m\lfloor n/m \rfloor + n \bmod m$$

where the first term is the quotient and the second the remainder.

Modular Arithmetic

- Another way of putting this is:
  - Given any positive integer n and any integer m, if we divide m by n, we get an integer quotient, q, and integer remainder, r, that obey the following relationship:

$$m = qn + r \qquad (0 \le r < n;\ q = \lfloor m/n \rfloor)$$

  - The remainder, r, is often referred to as a ***residue*** of m modulo n, and is the smallest non-negative integer that differs from m by a multiple of n.

For example,

$m = 11;\quad n = 7;\quad 11 = 1\times 7 + 4 \qquad r = 4$

$m = -11;\quad n = 7;\quad -11 = (-2)\times 7 + 3 \qquad r = 3$

Modular Arithmetic

- Two integers, a and b are said to be ***congruent*** (denoted by $\equiv$) if:

  $$a \equiv b \pmod{m} \iff a \bmod m = b \bmod m$$

  that is, "***a* is congruent to *b* modulo *m***"
- Alternatively, **in arithmetic modulo *m***, a and b are ***equivalent*** if their difference, (a - b), is a multiple of m; that is, $m \mid (a - b)$
- The set of integers $\boldsymbol{Z_m} = \{0,1, \ldots m - 1\}$ form **the complete set of residues modulo *m*** -- there are only m different integers, mod m
- The operation a mod m denotes the residue of a, such that the residue is some integer from 0 to m - 1. This operation is known as a ***modular reduction***.
  - Example: $10 \equiv 2 \pmod{4}$ because $4 \mid (10 - 2)$

Modular Arithmetic

- Congruence is an ***equivalence relation*** -- that is, it satisfies:

  1) The ***reflexive law*** : $a \equiv a$

  2) The ***symmetric law*** : $a \equiv b \Rightarrow b \equiv a$

  3) The ***transitive law*** : $a \equiv b \equiv c \Rightarrow a \equiv c$
- Finding the smallest non-negative integer to which k is congruent modulo n is called **reducing *k* modulo *n***

Modular Arithmetic

- We can also ***add*** and ***subtract*** congruent elements without losing congruence:

  $$a \equiv b \text{ and } c \equiv d \Rightarrow a + c \equiv b + d \pmod{m}$$
  $$\Rightarrow a - c \equiv b - d \pmod{m}$$

- ***Multiplication*** also works:

  $$a \equiv b \text{ and } c \equiv d \Rightarrow ac \equiv bd \pmod{m}, \text{ for integers } b, c$$

Modular Arithmetic

- Modular arithmetic is like ordinary arithmetic. It is:
  - ***commutative*** (for addition and multiplication)

    $a + b = b + a$
  - ***associative***

    $(a + b) + c = a + (b + c)$

  and
  - ***distributive***

    $a(b + c) = (ab) + (ac)$

    and

    $(b + c)a = (ba) + (ca)$

Modular Arithmetic

- A very important property of modular arithmetic is:
  - ***Reducing each intermediate result modulo m yields the same result as doing the entire calculation, and then reducing the result to modulo m:***

$$(a+b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$$
$$(a-b) \bmod m = ((a \bmod m) - (b \bmod m)) \bmod m$$
$$(a \cdot b) \bmod m = ((a \bmod m) \cdot (b \bmod m)) \bmod m$$
$$(a \cdot (b+c)) \bmod m = (((a \cdot b) \bmod m) + ((a \cdot c) \bmod m)) \bmod m$$

  - This means that we can do modular arithmetic without worrying about whether we will exceed some large arithmetic bound -- so such calculations can be done on computers, even for large integer values.

Modular Arithmetic

- Here are the possible values of $(a + b) \bmod 8$:

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

and $(a \cdot b) \bmod 8$:

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

Try a Java applet which demonstrates modular arithmetic

Modular Arithmetic: Exponentiation

- Recall that ***exponentiation*** is defined:

$$a^0 = e, \quad \text{the identity element}$$
$$a^n = a \bullet a \bullet \cdots \bullet a \quad (\text{i.e.} \bullet \text{ applied } n\text{-1 times})$$
$$a^{-n} = (a')^n, \quad \text{where } a' \text{ is the inverse of } a$$

  - In ordinary arithmetic, exponentiation rapidly produces very large numbers
  - However, because of the important property of modular arithmetic that intermediate results can be computed mod m, then is is possible in mod m arithmetic to do powerful exponentiation without producing very large numbers
  - Remember, in cryptography, we'll be dealing with very large values of m, so this is important.

Modular Arithmetic: Exponentiation

- For example, instead of performing the calculation:

$$a^n \bmod m = (a \bullet a \bullet a \bullet a \bullet \cdots \bullet a)$$

  we can instead perform fewer multiplications and use intermediate modular reductions.
  - Let's take a specific case of $a^8 \bmod n$. We can calculate it:

$$a^8 \bmod m = ((a^2 \bmod m)^2 \bmod m)^2 \bmod m$$

  - Similarly:

$$a^{16} \bmod m = (((a^2 \bmod m)^2 \bmod m)^2 \bmod m)^2 \bmod m$$
$$a^{25} \bmod m = (((((((a^2 \bmod m) \bullet a) \bmod m)^2 \bmod m)^2 \bmod m)^2 \bmod m) \bullet a) \bmod m$$

Modular Arithmetic: Division

- So far, for mod *m* arithmetic, we have addition, subtraction (defined through an additive inverse), and multiplication.
- What about ***division***?
  - Division is defined through a *multiplicative inverse*.
  - In regular arithmetic:
    - The multiplicative inverse of 5 is 1/5, because 5·1/5 = 1
  - In modular arithmetic, things are not so easy:

    Find x, where $5 \cdot x \equiv 1 \pmod 7$

    which is equivalent to finding an x and a k (both integers) such that:

    $5x = 7k + 1$
  - The general problem is to find x such that:

    $1 = (a \times x) \bmod m$

    or :

    $a^{-1} \equiv x \pmod m$

Modular Arithmetic: Multiplicative Inverse

- Sometimes the modular multiplicative inverse has a solution, and sometimes it doesn't:
  - The inverse of 5, mod 14, is 3

    5*3 mod 14 = 1
  - The inverse of 2, mod 14, doesn't exist.
    - Look at the row for 2, at right;
    - It does not contain a value 1
- It turns out that $a^{-1} \equiv x \pmod m$ **has a solution iff *a* and *n* are relatively prime**.
  - For example, look at the rows to the right.
  - The only rows that contain a 1 are for values that are relatively prime to 14:

    1, 3, 5, 9, 11, 13

The mod 14 multiplication table.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13
2	0	2	4	6	8	10	12	0	2	4	6	8	10	12
3	0	3	6	9	12	1	4	7	10	13	2	5	8	11
4	0	4	8	12	2	6	10	0	4	8	12	2	6	10
5	0	5	10	1	6	11	2	7	12	3	8	13	4	9
6	0	6	12	4	10	2	8	0	6	12	4	10	2	8
7	0	7	0	7	0	7	0	7	0	7	0	7	0	7
8	0	8	2	10	4	12	6	0	8	2	10	4	12	6
9	0	9	4	13	8	3	12	7	2	11	6	1	10	5
10	0	10	6	2	12	8	4	0	10	6	2	12	8	4
11	0	11	8	5	2	13	10	7	4	1	12	9	6	3
12	0	12	10	8	6	4	2	0	12	10	8	6	4	2
13	0	13	12	11	10	9	8	7	6	5	4	3	2	1

Modular Arithmetic: Multiplicative Inverse

- One way of finding the inverse of *a* modulo *m* is to extend Euclid's greatest common divisor algorithm:
  - The ***Extended Euclidean Algorithm:***
    - While computing gcd(*a*, *m*), we can also find two integers *u* and *v* such that:

      gcd(*a*, *m*) = *ua* + *vm*
    - If *a* and m are relatively prime, then the gcd(*a*, *m*) = 1, and:

      $1 = ua + vm = ua \pmod m$ (performing a reduction mod *m*)

      and then, multiplying both sides by a^{-1}:

      $a^{-1} = ua \cdot a^{-1} = u$
    - So, if gcd(*a*, *m*) = 1, then *u* is the multiplicative inverse of *a* mod *m*; otherwise, there is no multiplicative inverse

Finite, or Galois Fields

- A ***finite field*** (also known as a ***Galois* Field***) is a field with a finite number of elements. Finite fields are critical to the success of many cryptographic algorithms.
  - The finite fields are ***completely known:***
    - It can be shown that the order of a finite field (number of elements in the field) must be a power of a prime, p^n, where *n* is a positive integer.
- **For a given prime, *p*, the finite field of order *p*, GF(*p*) is defined as the set Z_p of integers {0, 1, ... , *p* - 1}, together with the arithmetic operations modulo *p*.**

***Evariste Galois (1811-1832),** French mathematician

Modular Arithmetic

- Here are the values for (a + b) mod 2:

and (a·b) mod 2:

Notice anything?

$$(a+b) \bmod 2 \Leftrightarrow a \texttt{ XOR } b$$
$$(a \cdot b) \bmod 2 \Leftrightarrow a \texttt{ AND } b$$

Try a Java applet which demonstrates modular arithmetic

Modular Arithmetic

- Another useful feature of arithmetic mod 2 is:
  - In the field $\mathbf{Z}_2$, ($\{0, 1\}$), there is only one inversion possible:

    1/1 = 1

    so division is the same operation as multiplication!

- Not surprisingly, the field $\mathbf{Z}_2$ is an important tool to analyze certain cryptographic algorithms by computer.

Modular Arithmetic

- Cryptography uses modular arithmetic a great deal, because:
  - Calculating discrete logarithms and square roots mod *n* can be hard problems.
  - It's easier to work with on computers, because it restricts the range of all intermediate values and results
    - For a *k*-bit modulus, *n*, the intermediate results of any addition, subtraction, or multiplication will not exceed 2*k* bits in length.
    - We can perform modular exponentiation without generating huge intermediate results
    - Arithmetic operations, mod 2, are natural for computers, because of the equivalence of addition with XOR, and multiplication with AND, etc.

$\mathbf{Z}_n^*$

- **Z** is the set of all integers
- We've seen that $\mathbf{Z}_n$ is the set of integers mod *n*
  - $\mathbf{Z}_{10} = \{0,1,2,3,4,5,6,7,8,9\}$

- $\mathbf{Z}_n^*$ is defined as the set of mod *n* integers that are relatively prime to *n*
  - $\mathbf{Z}_{10}^* = \{1,3,7,9\}$ (0 is missing because gcd(0, 10) = 10)

Z_n^*

- The multiplication table for Z_{10}^* provides some surprises:
- Notice anything?

	1	3	7	9
1	1	3	7	9
3	3	9	1	7
7	7	1	9	3
9	9	7	3	1

  - **Every element in Z_{10}^* is present in the table, and no other elements other than those are present. Furthermore, every element in Z_{10}^* is present in every row of the table.**

What does this mean for Z_{10}^*?

- It turns out that this is true for all n:
  Z_n^* is closed under multiplication mod n

Fermat's Little Theorem

- *Fermat's Little Theorem states:
  - If p is prime and a is a positive integer not divisible by p, then:

$$a^{p-1} \equiv 1 \bmod p$$

  - If we multiply both sides by a, we can come up with an alternative form:

$$a^{p-1} \cdot a \equiv a^p \equiv a \bmod p$$

***Pierre de Fermat (1601 - 1665)**, French mathematician most famous for *Fermat's Last Theorem*, which was considered one of mathematics' most difficult theorems, and has only recently been finally proven. His Little Theorem has nothing to do with his Last Theorem.

Euler's Totient Function

- An important quantity in number theory is ***Euler's Totient Function***:
  - **The number of positive integers less than n, that are relatively prime to n.**
- It is written $\varphi(n)$:

$$\varphi(1) = 1$$
$$\varphi(p) = p - 1 \qquad (\text{for } p \text{ prime})$$
$$\varphi(m) < m - 1 \qquad (\text{for } m \text{ composite})$$

- In other words, Euler's Totient Function $\varphi(n)$ is ***the number of elements in Z_n^****

***Leonhard Euler (1707 - 1783)**, Swiss mathematician

Euler's Totient Function

- Assume we have two distinct prime numbers, p and q, and an integer $n = pq$
  - Then:

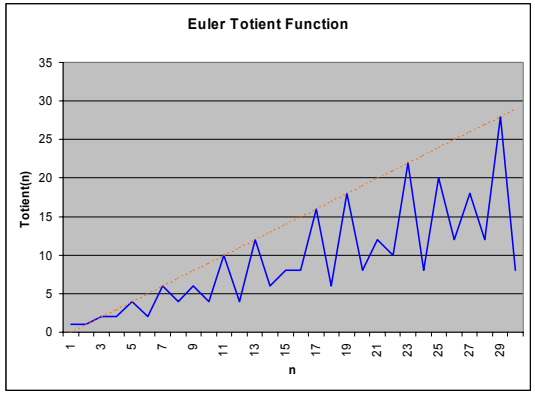
$$\varphi(n) = \varphi(pq) = \varphi(p) \times \varphi(q) = (p-1) \times (q-1)$$

  - The set of residues in $\boldsymbol{Z_n}$ is $\{0,1,\ldots,(pq - 1)\}$
  - The residues that are not relatively prime to n are:
    - The set $\{p, 2p, \ldots, (q - 1)p\}$, the set $\{q, 2q, \ldots, (p - 1)q\}$, and 0
  - So:

$$\begin{aligned}\varphi(n) &= pq - [(q-1) + (p-1) + 1] \\ &= pq - (p+q) + 1 \\ &= (p-1) \times (q-1) \\ &= \varphi(p) \times \varphi(q)\end{aligned}$$

Euler's Totient Function

- Here is what the Euler Totient Function values look like for small values of n. (The dotted red line is the line $f(n) = n - 1$)



Summary

- Whew! That's all the math we're going to do for now!
- We'll relate this to the cryptographic algorithms we're going to study in the near future.