The Algebra of Encryption

# Modular Arithmetic

Much of modern cryptography is based on modular arithmetic. We can think of modular arithmetic as a cycle. When we reach the base number or modulus, we cycle back to zero. We use modular arithmetic every day. For example, when we look at the clock we are using base 12. If the current time is 9 am and an appointment is at 1 pm, we know that the appointment is four hours away. Modular arithmetic tells us that .

## Division Algorithm

The division algorithm roughly states “that any integer a can be “divided” by b in such a way that the remainder is smaller than b.” Mathematically, this can be expressed as follows

Equation ‑

Where m is a multiplier and r is a residue.

Example ‑

, the multiplier is 1 and the residue is 1. Using modular arithmetic, we write .

Example ‑

, in this case the multiplier is zero and the residue is equal to 9 or a. Using modular arithmetic, we write .

## Addition

Addition follows the same rules that we are used to from normal arithmetic with the additional step that the result is reduced by the modulus.

To add two numbers and we first express the numbers in the form of Equation 1‑1.

Next we add and collect terms.

Note that this remains in the form of Equation 1‑1 but does not preclude the possibility that the sum could be greater than the modulus, thus causing a change to the multiplier.

Example ‑

In this case, the multiplier is not affected.

Example ‑

In this case, the resulting residue is greater than the modulus and should be reduced. Add and subtract 12, and collect terms.

## Subtraction

Modular subtraction is similar to modular addition. To subtract two numbers and we first express the numbers in the form of Equation 1‑1.

Next we subtract and collect terms.

Again, the result remains in the form of Equation 1‑1 but does not preclude the possibility that the difference could be less than the zero. Since we generally deal with positive numbers, this causes a change to the multiplier.

Example ‑

In this case, the multiplier is not affected.

Example ‑

In this case, the resulting residue is less than zero and should be adjusted. Add and subtract 12, and collect terms.

## Multiplication

Multiplication is merely repeated addition. As with addition and subtraction, the resulting residue is generally adjusted to be less than the modulus and non-negative.

Example ‑

In this case, the multiplier is not affected.

Example ‑

In this case, the resulting residue should be adjusted.

## Division

Division is a little tricky. Normally we think of division as dividing one number into equal parts countable to another. Thus the name. However, when we stick to Integers, we don’t always get equal Integer parts.

We need to think of modular division a little differently. Generally, when we divide c by d we use the following notation.

Equation ‑

We need to write this in a different way.

 or

Equation ‑

If trying to divide c by d, we need to ask by what number, e, can we multiply d such that the result is congruent to c modulo the modulus?

Example ‑

This is easy because we know we can multiply 2 by 5 and the result is 10.

Example ‑

In this case, . That is to say, we found a number, 9, that we can multiply by 4 such that the result is congruent to 10 modulo 13. Exactly how we find the number 9 directly, without trial and error, is discussed in section 1.6 and 0.

## Division by Multiplicative Inverse

Another way of accomplishing modular division is by using the multiplicative inverse of the divisor. Re-writing Equation 1‑2, we get the following equation for division.

Equation ‑

But this begs the question, what is the multiplicative inverse in modular arithmetic? Remembering back to regular arithmetic, we again think of multiplicative inverse in a different way. We need to ask by what number, , can we multiply d such that the result is congruent to 1 modulo the modulus? This is expressed mathematically as follows.

Equation ‑

Expressed algebraically for modular arithmetic, we have the following.

Equation ‑

Where again, m is the multiplier, b is the modulus, d is the number of interest, and is the modular multiplicative inverse (MMI).

Example ‑

.

Again, exactly how we find the number 10 directly, without trial and error, is discussed in section 0.

Example ‑

Going back to Example 1‑10, we get the following.

, which agrees with the findings in Example 1‑10.

# Useful Functions

## Euclidean Algorithm

The Euclidean algorithm is an important part of the algebra of encryption. It is generally thought of as providing the Greatest Common Divisor (GCD) of two numbers. When the two numbers are properly chosen, the Euclidean algorithm can also be used to directly determine the Modular Multiplicative Inverse (MMI).

### Greatest Common Divisor

“The Euclidean algorithm (also called Euclid’s algorithm) is an efficient method for computing the greatest common divisor.” “The GCD of two numbers is the largest number that divides both of them without leaving a remainder.”

Figure 2‑1 reproduces an example discussed in . We would like to find the GCD of two numbers represented by lines AB and CD.

1. Start by comparing the smaller number to the larger number.
2. Find the quotient of the two numbers. In this case, the quotient refers to the greatest integer less than or equal to . In other words, how many times can we multiply CD and still remain smaller than or equal to AB. In the figure, this quotient is 1.
3. Multiply the second number by the quotient and subtract from the first. In the figure, the result, or residue, is AE.
4. Now compare the residue with the previous smaller number. In the figure, this is comparing CD and AE. Repeat the above steps until the larger number is an integral multiple of the smaller number. That is to say, there is no resulting residue. The GCD is the last residue, in this case, CF.



Figure ‑: Euclidean Algorithm

Now we can add up the line lengths and verify that CF is the GCD of AB and CD.

Referring to Figure 2‑2, it is easy to see that CF does indeed evenly divide both AB and CD.



Figure ‑: Comparing the Results

### Modular Multiplicative Inverse

By computing the Euclidian algorithm on a number of interest and the modulus, we can directly compute the MMI of the number of interest. In these calculations, it is efficient to accumulate the quotients at each step as coefficients of the two numbers. This is often referred to as the Extended Euclidean algorithm.

If we put numbers to the lines in Figure 2‑1, for example 50 and 35, we have the following example.

Example ‑

1. Start by writing the two numbers as below.

50 = 50 ( 1) + 35 ( 0)

35 = 50 ( 0) + 35 ( 1)

2. Find the quotient of the two numbers.

q = int(50 / 35) = 1

3. Multiply the second equation through by the quotient and subtract from the first.

50 = 50 ( 1) + 35 ( 0)

35 = 50 ( 0) + 35 ( 1), q = 1

15 = 50 ( 1) + 35 ( -1)

4. Repeat steps 2 and 3 comparing the residues until the result is 0

50 = 50 ( 1) + 35 ( 0)

35 = 50 ( 0) + 35 ( 1), q = 1

15 = 50 ( 1) + 35 ( -1), q = 2

 5 = 50 ( -2) + 35 ( 3), q = 3

 0 = 50 ( 7) + 35 (-10)

The GCD is the last residue, in this case, 5. If we work this example again using the numbers 10 and 7, we will find that the GCD is 1. If two numbers have a GCD of 1, they are said to be relatively prime or coprime.

Now, how does this find the MMI? If we return to Example 1‑11, we wish to find the MMI of 4 modulo 13. If we compute the Extended Euclidian algorithm of 13 and 4 we have the following.

13 = 13 ( 1) + 4 ( 0)

 4 = 13 ( 0) + 4 ( 1), q = 3

 1 = 13 ( 1) + 4 ( -3)

When searching for the MMI, we can stop when the residue is 1. Now compare our result with Equation 1‑6. (-3) is the MMI of 4 modulo 13. We prefer to deal with non-negative numbers, so the result needs to be adjusted. We do this by adding and subtracting 13 \* 4 and collecting terms.

1 = 13 (1 ) + 4 (-3 ) + 13 (-4) + 4 (13)

1 = 13 (1 - 4) + 4 (-3 + 13)

1 = 13 ( -3) + 4 ( 10)

Thus 10 is the MMI of 4 mod 13. This result agrees with our findings in Example 1‑11.

Mentioned above is that we can stop searching for the MMI when the residue is 1. This is because the next line would result in a residue of zero. When we arrive at a residue of zero but the previous line does not have a residue of 1, the number of interest and the modulus are not coprime and no MMI exists for that number of interest for that modulus.

## Modular Exponentiation

Many encryption schemes employ modular exponentiation. When dealing with very large numbers, such as in cryptography, efficient computing is essential. describes an algorithm entitled Fast Modula Exponentiation.

* Initiate X = base, E = exponent, Y = 1: .
* If E is odd: replace Y by (X \* Y) mod m and replace E = E - 1.
* E is now even: replace X by (X \* X) mod m and replace E = E / 2.
* When E = 0: .

X keeps the current binary power of the base number. Y is the accumulator. Note how E is parsed for its binary bit values.

1. Starting from the least significant bit, check if the bit is 1 (if E is odd).
2. If E is odd, accumulate the corresponding power of the base: Y = Y \* X. The decrement of E simply makes the division by 2 in the next step easier.
3. E is now even, update X to the next binary power of the base: X = X \* X. Divide E by 2, that is to say, right shift so the next binary bit is in the least significant bit position.
4. Loop to step 2 until E = 0.

Example ‑

First let’s calculate without a modulus.

E = 11 = 8 + 2 + 1.

Y = 311 = 38 \* 32 \* 31 = 6561 \* 9 \* 3 = 177147

| Notes | X | E | Y |
| --- | --- | --- | --- |
| Initialization |  | 11 | 1 |
| E is odd |  |  |  |
| E is even |  |  |  |
| E is odd |  |  |  |
| E is even |  |  |  |
| E is even |  |  |  |
| E is odd |  |  |  |

Example ‑

Now let’s calculate

Y = 311 = 38 \* 32 \* 31 = 237 \* 9 \* 3 mod 527 = 75 mod 527

| Notes | X | E | Y |
| --- | --- | --- | --- |
| Initialization |  | 11 | 1 |
| E is odd |  |  |  |
| E is even |  |  |  |
| E is odd |  |  |  |
| E is even |  |  |  |
| E is even |  |  |  |
| E is odd |  |  |  |

You can verify that .

Notice that everything happens the same with a modulus as without a modulus. The only difference is in the size of the numbers - the calculation results are reduced modulo 527.

Also notice the significant role of multiplication in this algorithm. We multiply X\*X for each exponent bit position. We multiply X\*Y when the exponent’s least significant bit is 1. In Example 2‑2 we dealt with larger numbers. In Example 2‑3 the numbers were limited by the modulus and remained smaller.

Consider multiplying two 4 digit binary numbers and two 2 digit binary numbers.

 1111

 x 1111

-----------------

 1111

 1111

 1111

 + 1111

-----------------

 11100001 11

 x 11

-----------------

 11

 + 11

-----------------

 1001

The 4 digit multiplication involved the addition of 16 bits. The 2 digit multiplication involved the addition of only 4 bits. Since the size of numbers cryptography deals with are usually larger than a computer’s arithmetic logic unit can deal with, special Big Integer computational packages such as Multiple Precision Integers and Rationals (MPIR) (MPIR home page) or Gnu Multiple Precision (GMP) (The GNU Multiple Precision Arithmetic Library) are usually employed. The smaller the modulus, the smaller the numbers we deal with, and the less time involved in multiplying.

## Chinese Remainder Theorem (CRT)

Chinese Remainder Theorem is frequently used in cryptography to reduce the calculation time inherent in calculating with large numbers. This time reduction is accomplished by doing a few extra calculations, but with much smaller numbers. Also, a significant part of the calculations may be done once, in advance, and then used for subsequent calculations.

### Pre-calculations

This discussion and example follows (Stallings, 2011, pp. p 254-257):

#### Choose the Factors of M

Choose and . The numbers must be coprime.

#### Calculate

Calculate for each .

To continue the example:

Note that this is NOT a simple swapping of and . The example only gives that appearance because we are using two factors. If M had three or more factors, this would be more clear.

#### Calculate

Use the Extended Euclidean Algorithm to calculate for each .

To continue the example:

#### Calculate

For convenience, define for each .

In the example:

### Addition

To add two numbers, complete the addition for each , then combine the results.

#### Add for each

Compute for each .

For example (Stallings, 2011, pp. p 254-257), to add 973 and 678 mod 1813:

+

---------------------------------

+

---------------------------------

#### Combine the results

Find the result of

To complete the example:

### Multiplication

To multiply two numbers, complete the multiplication for each , then combine the results.

#### Multiply for each

Compute for each .

For example (Stallings, 2011, pp. p 254-257), to multiply 1651 and 73 mod 1813:

x

---------------------------------

x

---------------------------------

#### Combine the results

Find the result of

To complete the example:

## Euler’s Totient Function

Euler’s totient function, , identifies the number of integers, less than n, that are relatively prime to n. A good treatment of Euler’s Totient function can be found in (Burton, 2007, pp. 131-135).

 presents Euler’s totient function as for prime p. This is correct, but not good for a general definition of the function. (Burton, 2007, pp. 131-135) gives a more versatile definition as . It is easy to see that this version degenerates to the previous version in the case of .

To determine the totient of a composite number, we must know the prime factors of the number. Given , (Burton, 2007, pp. 131-135). Again, if and , then the equation degenerates to as described by .

Example ‑

To calculate , the first form is adequate. First we must know the prime factors of 21, namely 3 and 7.

Therefore, the number of integers less than 21 relatively prime to 21 should count to 12.

1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20 are the 12 numbers less than 21 that are relatively prime to 21.

Example ‑

To calculate , the second form of the totient function is required. The prime factors of 20 are and 5.

The 8 integers less than 20 relatively prime to 20 are 1, 3, 7, 9, 11, 13, 17, 19.

# Public Key Cryptography - RSA

Euler’s theorem states that if and , then . (Burton, 2007, p. 137) This is the heart of the RSA encryption and decryption method. (RSA, 2011).

If we choose two prime numbers p and q we can form and determine . If we then choose an encryption exponent e so that , we can find a decryption exponent d such that . The exponent d is merely the MMI of , and we know from section 2.1.2 how to find d. Note that for some multiplier m.

Now take a message expressed as a number M. Calculate the encrypted message by using the encryption exponent.

To decrypt, use the decryption exponent. Calculate .

Taking note of Euler’s theorem, if the decryption calculations are made modulo n, we have

Thus we have successfully decrypted the message, assuming to satisfy the requirements of Euler’s theorem. “In the unlikely event that M and n are not relatively prime, a similar argument establishes that and , which then yields the desired congruence . We omit the details.” (Burton, 2007, p. 204)

RSA involves calculating modular exponentials. The example uses normal encryption and CRT decryption. Note that CRT only benefits decryption because the factors of the modulus must be known and they are part of the private key.

Let:

message:

calculate

calculate

calculate

## CRT Pre-calculations

## Encrypt

 (See Example 2‑3.)

## Decrypt

Note the use of and in place of the normal decrypt exponent. These exponents are d modulus and , NOT d modulus p and q.

## Combine the results

## Conclusion

Decryption is accomplished using smaller exponents and smaller moduli. Multiplication is a large part of calculating modular exponentials. When computing many exponentials using the same private key, time savings becomes significant.

# How to Share a Secret

 describes how to share secret information such that the holders of a share cannot individually obtain the secret. Only if a certain minimum number of share holders collaborate can the secret information be identified.

A simple description of this method is to use a curve described by a polynomial function.

Equation ‑

Typically, is the secret information, for example a secret key, and the other t coefficients in the polynomial may be chosen at random. If two or more elements comprise the secret key, such as , necessary for CRT decryption of RSA, then more coefficients may be set and the remainder chosen at random.

Basic algebra tells us that when we have unknowns - the coefficients, we need to know points on the curve to identify all of the coefficients. We really only want the one coefficient, , but even that cannot be determined without the required minimum number of points from the curve.

Points on the curve - x and f(x) pairs, are the secret shares given to the trusted authorities. Many more than secret shares may be given out. But, only if a minimum of trusted authorities agree to provide their secret share and collaborate, can the secret information be identified. The secret information - coefficient(s) of the polynomial, may be identified by any linear algebra method for solving a system of simultaneous equations.

The x values may be as simple as indexes, incremented between one share holder to the next. Only the value of f(x) must be held secret.

# Paillier Cryptography

The cryptographic systems proposed by involve polynomial expansion of exponentiation. Paillier proposed three systems, one will be discussed here. It is similar to RSA cryptography.

## Carmichael Function

In order to understand the algebra of Paillier cryptography, we need to understand the Carmichael function. The Carmichael function, , is similar to Euler’s totient function .

Equation ‑

where , and are distinct primes, and lcm stands for Least Common Multiple.

The Carmichael function also has some useful properties for Paillier cryptography. For any integer that is coprime with :

Equation ‑

Which implies there exists some integer and such that

Equation ‑

## Preliminaries

Choose two safe prime numbers p and q, form , and determine . Safe prime numbers are of the form , where p is also a prime. (Safe prime, 2010)

Define the function

Equation ‑

Choose a generator value g such that

Equation ‑

The public key is (g, n). The private key is λ.

## Encrypt

The plaintext message is . Choose a random number . Calculate the encrypted message

Equation ‑

## Decrypt

Plaintext

Equation ‑

## Explanation

**The Generator g**

Starting from the Carmichael function Equation 5‑3.

Equation ‑

If we use binomial expansion of the polynomial, we get (Binomial theorem, 2011)

where

When we reduce modulo , all of the polynomial terms of and higher drop out. We are left with

Equation ‑

**The Decrypt Numerator**

Applying Equation 5‑2.

Equation ‑

Applying Equation 5‑9 to

**The Decrypt Denominator**

Equation ‑

**The Decrypt**

Combining Equation 5‑10 and Equation 5‑11 into Equation 5‑7, we get

## Blinding

Cryptographic blinding allows for a message to be multiplied by a specially treated random number, while still allowing the message to be decrypted without knowledge of the random number.

Due to the properties of Paillier cryptography noted in Equation 5‑2, any succession of blinding factors may be applied to the ciphertext without affecting the successful decryption.

To see this, apply a succession of blinding factors to the same message.

where . is still a random number and does not affect the successful decryption of the message.

## Additive Homomorphic Properties

Paillier cryptography includes an interesting homomorphic property in that the multiplication of two ciphertexts is equivalent to the addition of the respective plaintexts.

To see this, start with two messages and and encrypt.

Now multiply the two ciphertexts.

where and . is a random number, and the product will correctly decrypt as .

## Pre-calculations

Some elements of Paillier cryptography can be pre-calculated.

**Decryption**

All of the elements of the decrypt denominator are known to the decryptor and may be pre-calculated.

Equation ‑

This is of course an MMI modulo .

Decryption then becomes

Equation ‑

**Determine λ**

Because p and q are safe primes, we have the following determination for λ.

where and are also primes.

This leads to the following pre-calculation for λ.

Equation ‑

## Observations

**Choosing g with a common factor**

The choosing of g, Equation 5‑5, allows a vulnerability. If shares a factor with , the division of the numerator, , by the denominator, , does not always yield an integer. If this goes un-noticed, and the result is truncated, the equality of Equation 5‑5 may hold and the value of will be accepted.

The public key is . If shares a factor with , the astute adversary would be able to employ the Euclid algorithm to factor n and thereby find the secret key .

**Choosing g without Blinding**

If one should choose , for some integer , blinding is in fact necessary. Consider encryption Equation 5‑6 without the blinding factor .

Following the same process as the derivation of Equation 5‑9, we arrive at

Because (g, n) is the public key, the value of a is known publicly. Thus m can be easily calculated from c. Thus Blinding is necessary if is chosen such that .

# Works Cited

*Binomial theorem.* (2011, June 21). Retrieved July 10, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Binomial\_expansion

*Blinding (cryptography).* (2011, June 3). Retrieved July 10, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Blinding\_(cryptography)

Burton, D. M. (2007). *Elementary Number Theory, Sixth Edition.* New York, New York 10020: McGraw-Hill Higher Education.

*Chinese remainder theorem.* (2011, July 2). Retrieved July 7, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Chinese\_remainder\_theorem

*Coprime.* (2011, June 25). Retrieved July 7, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Coprime

*Euclidean algorithm.* (2011, June 30). Retrieved July 7, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Euclid\_algorithm

Garrett, P. (2004). *The Mathematics of Coding Theory.* Upper Saddle River, New Jersey: Pearson Prentice Hall.

*Modular arithmetic.* (2011, July 5). Retrieved July 7, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Modular\_arithmetic

*MPIR home page*. (n.d.). Retrieved July 9, 2011, from MPIR: http://www.mpir.org/

Paillier, P. (1999). Public-Key Cryptosystems Based on Composite Degree Residuosity Clases. *Advances in Cryptology - Eurocrypt '99* , pp. 223-238.

*RSA.* (2011, July 8). Retrieved July 8, 2011, from Wikipedia: http://en.wikipedia.org/wiki/RSA

*Safe prime.* (2010, August 24). Retrieved July 9, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Safe\_prime

Shamir, A. (November, 1979). How to Share a Secret. *Communications of the ACM* , 612-613.

Stallings, W. (2011). *Cryptography and Network Security, Principles and Practice, Fifth Edition.* Prentice Hall.

*The GNU Multiple Precision Arithmetic Library*. (n.d.). Retrieved July 9, 2011, from GNU: http://gmplib.org/