Modular Arithmetic

Scott Contini[1], Çetin Kaya Koç[2], Colin D. Walter[3]
[1]Silverbrook Research, New South Wales, Australia
[2]College of Engineering and Natural Sciences, Istanbul Sehir University, Uskudar, Istanbul, Turkey
[3]Information Security Group, Royal Holloway University of London, Surrey, UK

Synonyms

Residue arithmetic

Related Concepts

►Finite Field; ►Prime Fields; ►Residue; ►Rings

Definition

Modular arithmetic is almost the same as the usual arithmetic of whole numbers. The main difference is that operations involve remainders after division by a specified number (the *modulus*) rather than the integers themselves.

Background

Modular arithmetic is a key ingredient of many public key cryptosystems. It provides finite structures (called "rings") which have all the usual arithmetic operations of the integers and which can be implemented without difficulty using existing computer hardware. An important property of these structures is that they appear to be randomly permuted by operations such as exponentiation, but the permutation is often easily reversed by another exponentiation. For suitably chosen cases, these operations enable *encryption* and decryption or *signature generation* and verification. Direct applications include *RSA public-key encryption* and the *RSA digital signature scheme* [17], *ElGamal public key encryption* and the *ElGamal digital signature scheme* [3], the *Fiat-Shamir signature scheme* [4], the *Schnorr Identification Protocol* [18], and *Diffie-Hellman key agreement* [2].

Modular arithmetic is also used to construct *finite fields* and in tests during *prime generation* [7] (►Probabilistic Primality Test). Several copies of the modular structures form higher dimensional objects in which lines, planes, and curves can be constructed. These can be used to perform *elliptic curve cryptography* (ECC) [6, 12] and to construct *threshold schemes* [19]. Additionally, modular arithmetic is used in some *hash functions* and *symmetric key* primitives. In many such cases, the modulus is implied by the computer word size, but other times the modulus is explicitly stated.

Theory

Introduction

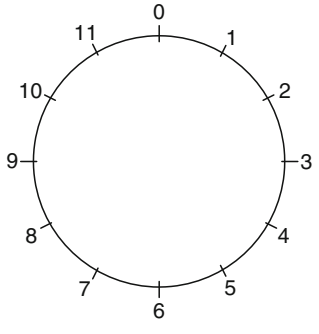
There are many examples of modular arithmetic in everyday life. It is applicable to almost any measurement of a repeated, circular or cyclic process. Clock time is a typical example: seconds range from 0 to 59 and just keep repeating, hours run from 0 to 11 (or 23) and also keep repeating, days run from Sunday (0, say) to Saturday (6, say). These are examples of arithmetic modulo 60, 12 (or 24), and 7, respectively. Measuring angles in degrees uses arithmetic modulo 360.

To understand arithmetic in modulus N, imagine a line of length N units, where the whole number points $0, \ldots, N-1$ are labelled. Now connect the two end points of the line so that it forms a circle of circumference N. Performing modular arithmetic with respect to *modulus* N is equivalent to arithmetic with the marked units on this circle.

An example for $N = 12$ is shown in Fig. 1. If one starts at number 0 and moves 14 units forward, the number 2 is reached. This is written $14 = 2 \pmod{12}$. Similarly, one can walk backwards 15 units from 0 and end up at 9. Hence, $-15 = 9 \pmod{12}$. In this arithmetic, every 12 is discarded. Equivalently, for any two numbers A and B such that $A = B \pmod{12}$, 12 divides the difference $A - B$.

Modular addition is the same as addition of units on this circle. For example, if $N = 12$ and the numbers 10 and 4 are added on this circle, the result is 2. This is because if one starts at position 10 and moves ahead 4 units, position 2 is reached. So four hours after 10 o'clock is 2 o'clock. This is written $10+4 = 2 \pmod{12}$. The result is the remainder (or "residue") after division by 12, i.e., $10 + 4 = 14$ becomes 14–12, namely 2.

The notation for modular arithmetic is almost identical to that for ordinary (integer) arithmetic. The main difference is that most expressions and equations specify



Modular Arithmetic. Fig. 1 Geometric view of arithmetic modulo 12

the modulus. Thus,

$$14 = 2 \pmod{12}$$

states that 14 and 2 represent the same element in a set which is called the *ring of residues mod 12*. When the modulus is clear, it may be omitted, as in

$$14 \equiv 2$$

The different symbol ≡ is needed because 14 and 2 are not equal as integers. The equation (or "congruence") is read as "14 is congruent to 2." All the integers in the set $\{\ldots, -22, -10, 2, 14, 26, \ldots\}$ represent the same *residue class* (or *congruence class*) modulo 12 because they all give the same remainder on division by 12, *i.e* the difference between any two of them is a multiple of 12. In general, the numbers $A, A+N, A+2N, A+3N, \ldots$ and $A-N, A-2N, A-3N, \ldots$ are all *equivalent* modulo N. Normally one works with the least nonnegative representative of a class, 2 in this case, because of the convenience of the unique choice when equality is tested, and because it takes up the least space. (Note that some programming languages incorrectly implement the modular reduction of negative numbers by failing to take proper account of the sign. The Microsoft Windows calculator correctly reduces negatives, but gives the greatest nonpositive value, namely, −10 in the above example.)

Modular Arithmetic Operations

Addition, subtraction, and multiplication are performed in exactly the same way as for integer arithmetic. Strictly speaking, the arithmetic is performed on the residue classes but, in practice, integers are picked from the respective classes, and they are worked with instead. Thus,

$$7 \times 11 + 3 = 80 = 8 \pmod{12}$$

In the expression on the left, the least nonnegative residues have been selected for working with. The result, 80, then requires a modular reduction to obtain a least nonnegative residue. Any representatives could be selected to perform the arithmetic. The answer would always differ by at most a multiple of the modulus, and so it would always reduce to the same value.

Hardware usually performs such reductions as frequently as possible in order to stop results from overflowing. Optimising integer arithmetic to perform modular arithmetic is the subject of much research. Modular multiplication is one of the most important areas of value to those implementing cryptographic functions; another is modular exponentiation. Montgomery [13] and Barrett [1] have created the most widely used methods for modular multiplication (Montgomery Modular Arithmetic and Barrett Reduction). Such operations make data-dependent use of power. This makes their use in embedded cryptosystems (e.g., smart cards) susceptible to attack through timing variations [8], compromising emanations [15], and differential power analysis [9] (Timing Attack, RF Attack and Smartcard Tamper Resistance). Secure implementation of modular arithmetic is therefore at least as important as efficiency in such systems.

Addition, subtraction, and multiplication behave in the same way for residues as for integer arithmetic. The usual identity, commutative and distributive laws hold, so that the set of residue classes form a "ring" in the mathematical sense, denoted $\mathbb{Z}_N$ for modulus N. Thus,

- $N \equiv 0 \pmod{N}$.
- $A + 0 \equiv A \pmod{N}$.
- $1 \times A \equiv A \pmod{N}$.
- if $A \equiv B \pmod{N}$, then $B \equiv A \pmod{N}$.
- if $A \equiv B \pmod{N}$ and $B \equiv C \pmod{N}$, then $A \equiv C \pmod{N}$.
- if $A \equiv B \pmod{N}$ and $C \equiv d \pmod{N}$, then $A + C \equiv B + d \pmod{N}$.
- if $A \equiv B \pmod{N}$ and $C \equiv d \pmod{N}$, then $A \times C \equiv B \times d \pmod{N}$.
- $A + B \equiv B + A \pmod{N}$.
- $A \times B \equiv B \times A \pmod{N}$.
- $A + (B + C) \equiv (A + B) + C \pmod{N}$.
- $A \times (B \times C) \equiv (A \times B) \times C \pmod{N}$.
- $A \times (B + C) \equiv (A \times B) + (A \times C) \pmod{N}$.

However, division is generally a problem unless the modulus is a prime. Since

$$10 = 2 \times 5 = 2 \times 11 \pmod{12}$$

it is clear that division by 2 (mod 12) can produce more than one answer; it is not uniquely defined. In fact, division by 2(mod 12) is not possible in some cases: $2x \pmod{12}$ always gives an even residue, so 3(mod 12) cannot be divided by 2. It can be shown that division by $A \pmod{N}$ is always well-defined precisely when A and N share no common factor, i.e., when they are *co-prime*. Thus, division by 7 is possible in modulo 12, but not division by 2 or 3.

If 1 is divided by 7(mod 12), the result is the *multiplicative inverse* of 7. Since $7 \times 7 = 1 \pmod{12}$, 7 is its own inverse. Following the usual notation of real numbers, this inverse is written 7^{-1}. For large numbers, the extended *Euclidean algorithm* [5] is used to compute multiplicative inverses. More precisely, to find the inverse of $A \pmod{N}$, one inputs the pair A, N into the algorithm, and it outputs X, Y such that $A \times X + N \times Y = \gcd(A, N)$, where gcd is the *greatest common divisor*. If the gcd is 1, then X is the inverse of $A \pmod{N}$. Otherwise, no such inverse exists.

Modular exponentiation (►Exponentiation Algorithms) is the main process in many of the cryptographic applications of this arithmetic. The notation is identical to that for integers and real numbers. $C^D \pmod N$ is D copies of C all multiplied together and reduced modulo N. As mentioned, the multiplicative inverse is denoted by an exponent $^{-1}$. Then the usual power laws, such as $x^A \times x^B = x^{A+B} \pmod N$, hold in the expected way.

When a composite modulus is involved, say N, it is often easier to work modulo its factors. Usually a set of co-prime factors of N is chosen such that the product is N. Solutions to the problem for each of these factors can then be pieced together into a solution modulo N using the *Chinese Remainder Theorem* (CRT) [14]. Implementations of the RSA cryptosystem which store the private key can use CRT to reduce the workload of decryption by a factor of 4.

An interesting aside is that the ring of integers modulo 0, i.e., $\mathbb{Z}_0$, is just the usual set of whole numbers with its normal operations of addition and multiplication: two whole numbers which belong to the same residue class must differ by a multiple of 0, and so have to be equal.

Multiplicative Groups and Euler's ϕ Function

The numbers which are *relatively prime* to (or just "prime to" for short) the modulus N have multiplicative inverses, as noted above. So they form a group under multiplication. Consequently, each number X which is prime to N has an *order* mod N which is the smallest positive integer n such that $X^n = 1 \pmod N$. The Euler phi function ϕ gives the number of elements in this group, and it is a multiple of the order of each element. So $X^{\phi(N)} = 1 \pmod N$ for X prime to N, and, indeed, $X^{k\phi(N)+1} = X \pmod N$ for such X and any k. This last is essentially what is known as Euler's Theorem. As an example, $\{1, 5, 7, 11\}$ is the set of residues prime to 12. So these form a multiplicative group of order $\phi(12) = 4$ and $1^4 = 5^4 = 7^4 = 11^4 = 1 \pmod{12}$. A special case of this result is *Fermat's "little" theorem* which states that $X^{P-1} = 1 \pmod P$ for a prime P and integer X which is not divisible by P. These are really the main properties that are used in reducing the cost of exponentiation in cryptosystems and in probabilistic primality testing (►Miller-Rabin Probabilistic Primality Test) [11, 16].

When $N = PQ$ is the product of two distinct primes P and Q, $\phi(N) = (P-1)(Q-1)$. RSA encryption on plaintext M is performed with a public exponent E to give ciphertext C defined by $C = M^E \pmod N$. Illustrating this with $N = 35$, $M = 17$ and $E = 5$, the computation is $C \equiv 17^5 \equiv (17^2)^2 \times 17 \equiv 289^2 \times 17 \equiv 9^2 \times 17 \equiv 81 \times 17 \equiv 11 \times 17 \equiv 187 \equiv 12 \pmod{35}$. The private decryption exponent D must have the property that $M = C^D \pmod N$, i.e., $M^{DE} = M \pmod N$. From the above, the value of D must satisfy $DE = k\phi(N) + 1$ for some k, i.e., D is a solution to $DE \equiv 1 \bmod (P-1)(Q-1)$. A solution is obtained using the *Euclidean algorithm* [5]. For the example, $D = 5$ since $\phi(35) = 24$ and $DE \equiv 5 \times 5 \equiv 1 \pmod{24}$. So $M \equiv 12^5 \equiv (12^2)^2 \times 12 \equiv 144^2 \times 12 \equiv 4^2 \times 12 \equiv 192 \equiv 17 \pmod{35}$, as expected. RSA chooses moduli which are products of two (large) primes so that decryption works also for texts which are not prime to the modulus. A nice exercise for the reader is to prove that this is really true. CRT is useful in the proof.

Prime Fields

When the modulus is a prime P, every residue except 0 is prime to the modulus. Hence, every nonzero number has a multiplicative inverse. So residues mod P form a *field* with P elements, written $\mathbb{F}_P$ or $GF(P)$. These *prime* fields are examples of *finite fields* [10]. The smallest such field is $\mathbb{F}_2$ which contains the two values 0 and 1. Because every nonzero has an inverse, the arithmetic of these fields is similar in many ways to that of the real numbers, and it is possible to perform similar geometric constructions. They already form a very rich source for cryptography, such as *Diffie-Hellman key agreement* [2] and *elliptic curve cryptography* [6, 12], and will undoubtedly form the basis for many more cryptographic primitives in the future.

Recommended Reading

1. Barrett P (1987) Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In: Odlyzko AM (ed) Advances in Cryptology – CRYPTO '86, Lecture Notes in Computer Science, vol 263. Springer, New york, pp 311–323. http://www.springerlink.com/content/c4f3rqbt5dxxyad4/
2. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inform Theory 22(6):644–654. http://citeseer.ist.psu.edu/diffie76new.html
3. ElGamal T (1985) A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inform Theory 31(4):469–472. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1057074
4. Fiat A, Shamir A (1987) How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko AM (ed) Advances in Cryptology – CRYPTO '86, Lecture Notes in Computer Science, vol 263. Springer, Berlin, pp 186–194. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.8796
5. Knuth DE (1998) The art of computer programming, vol 2, Seminumerical Algorithms, 3rd edn. Addison-Wesley, Reading, ISBN 0-201-89684-2. http://www.informit.com/title/0201896842
6. Koblitz N (1987) Elliptic curve cryptosystems. Math Comput 48(177):203–209. http://www.jstor.org/pss/2007884
7. Koblitz N (1994) A course in number theory and cryptography, Graduate Texts in Mathematics, vol 114, 2nd edn. Springer, New York, ISBN 978-0-387-94293-3. http://www.springer.com/math/numbers/book/978-0-387-94293-3
8. Kocher P (1996) Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz N (ed)

Advances in Cryptology – CRYPTO '96, Lecture Notes in Computer Science, vol 1109, Springer, Berlin, 104–113. http://www.springerlink.com/content/4el17cvre3gxt4gd/
9. Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Wiener M (ed) Advances in Cryptology – CRYPTO '99, Lecture Notes in Computer Science, vol 1666, Springer, Berlin, pp 388–397. http://www.springerlink.com/content/kx35ub53vtrkh2nx/
10. Lidl R, Niederreiter H (1994) Introduction to finite fields and their applications, 2nd edn. Cambridge University Press, Cambridge, ISBN 9780521460941. http://www.cambridgeuniversitypress.com/catalogue/catalogue.asp?isbn=9780521460941
11. Miller GL (1976) Riemann's hypothesis and tests for primality. J Comput Syst Sci 13(3):300–317. http://www.cs.cmu.edu/~glmiller/Publications/b2hd-Mi76.html
12. Miller V (1986) Uses of elliptic curves in cryptography. In: Williams HC (ed) Advances in Cryptology – CRYPTO '85: Proceedings, Lecture Notes in Computer Science, vol 218, Springer, Berlin, pp 417–426. http://www.springerlink.com/content/4lfhkd08684v3wyl/
13. Montgomery PL (1985) Modular multiplication without trial division. Math Comput 44(170):519–521. http://www.jstor.org/pss/2007970
14. Quisquater J-J, Couvreur C (1982) Fast decipherment algorithm for RSA Publickey cryptosystem. Electron Lett 18(21):905–907. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4246955
15. Quisquater J-J, Samyde D (2001) ElectroMagnetic analysis (EMA): measures and counter-measures for smart cards. In: Attali I, Jensen T (eds) Smart card programming and security (e-Smart 2001), Lecture Notes in Computer Science, vol 2140, Springer, Cannes, pp 200–210. http://www.springerlink.com/content/chmydkq8x5tgdrce/
16. Rabin MO (1980) Probabilistic algorithm for testing primality. J Number Theory 12(1):128–138. http://dx.doi.org/10.1016/0022-314X(80)90084-0
17. Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.5588
18. Schnorr CP (1991) Efficient signature generation by smart cards. J Cryptol 4:161–174. http://www.springerlink.com/content/w037127811042441/
19. Stinson DR (2005) Cryptography: theory and practice, 3rd edn. CRC Press, New York, ISBN 9781584885085. http://www.crcpress.com/product/isbn/9781584885085

Modular Root

Scott Contini
Silverbrook Research, New South Wales, Australia

Related Concepts

►Euler's Totient Function; ►Integer Factoring; ►Modular Arithmetic; ►Number Theory; ►Quadratic Residue; ►RSA Problem

Definition

In the congruence $x^e \equiv y \bmod n$, x is said to be the e^{th} *modular root* of y with respect to modulus n.

Background

The cases that are of interest to cryptography have $\gcd(x, n) = \gcd(y, n) = 1$. Algorithms for finding modular roots are relevant to the security of the ►RSA cryptosystem.

Theory

Computing modular roots is no more difficult than finding the order of the multiplicative group modulo n. In number theoretic terminology, this value is known as ►Euler's totient function, $\phi(n)$, which is defined to be the number of integers in $\{1, 2, \ldots, n-1\}$ that are ►relatively prime to n. If $\gcd(e, \phi(n)) = 1$, then there is either one or zero solutions, depending upon whether y is in the multiplicative ►subgroup generated by x. Assuming it is, the solution is obtained by raising both sides of the congruence to the power $e^{-1} \bmod \phi(n)$. If the gcd condition is not 1, then there may be more than one solution. For example, consider the special case of $e = 2$ and n an odd integer larger than 1. The congruence can have solutions only if y is a ►quadratic residue modulo n. Furthermore, if x is one solution, then $-x$ is another, implying that there are at least two distinct solutions.

Open Problems

Computing modular roots is easy when n is prime since then $\phi(n) = n - 1$. The more interesting case is when n is composite, where it is known as the ►RSA problem. An important open question is whether a method exists for computing modular roots faster than ►integer factoring. Note that any method which finds $\phi(n)$ cannot be faster than factoring since determining $\phi(n)$ is provably as difficult as factoring n.

Modulus

Scott Contini
Silverbrook Research, New South Wales, Australia

Related Concepts

►Modular Arithmetic; ►Number Theory

Definition

In ►modular arithmetic, the operand that the mod operation is computed with respect to is known as the *modulus*.