Modular Arithmetic

Scott Contini

Çetin K. Koç,
Istanbul Şehir University &
University of California Santa Barbara

Colin D. Walter,
Information Security Group,
Royal Holloway, University of London.

Synonyms

– Residue arithmetic

Related Concepts and Keywords

– Arithmetic
– Modulus
– Residue
– Remainder
– Modular multiplication
– Montgomery modular multiplication
– Modular exponentiation
– Prime fields
– Finite fields
– 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] (see also 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) [12,6] 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.

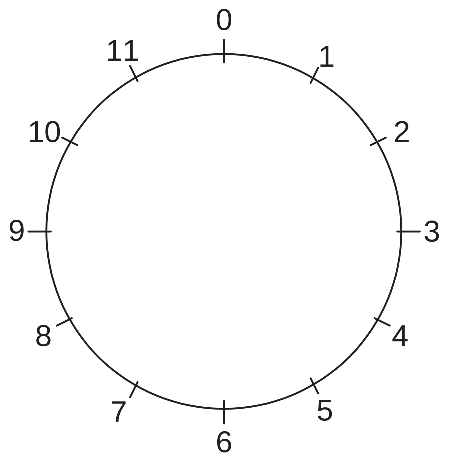

Fig. 1. Geometric view of arithmetic modulo 12.

An example for $N = 12$ is shown in Figure 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 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 $\equiv$ 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 $\{..., -22, -10, 2, 14, 26, ...\}$ 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 non-negative 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 non-positive value, namely −10 in our 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 non-negative residues have been selected for working with. The result, 80, then requires a modular reduction to obtain a least non-negative 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 (see also 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] (see also 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: $2x$ (mod 12) always gives an even residue, so 3 (mod 12) cannot be divided by 2. It can be shown that division by A (mod 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 (*see* 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 (see also 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 $DE \equiv 5\times5 \equiv 1 \pmod{24}$. So $M \equiv 12^5 \equiv (12^2)^2\times12 \equiv 144^2\times12 \equiv 4^2\times12 \equiv 192 \equiv 17 \pmod{35}$, 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 non-zero 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 non-zero 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 [12,6], and will undoubtedly form the basis for many more cryptographic primitives in the future.

Recommended Reading

[1] P. Barrett, "Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor", *Advances in Cryptology* – Crypto '86, A. M. Odlyzko (ed), Lecture Notes in Computer Science 263, pp. 311–323, Springer Verlag, 1987.
`http://www.springerlink.com/content/c4f3rqbt5dxxyad4/`

[2] W. Diffie and M. E. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory* 22(6), pp. 644–654, November 1976.
`http://citeseer.ist.psu.edu/diffie76new.html`

[3] T. ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Transactions on Information Theory* 31(4), pp. 469–472, July 1985.
`http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1057074`

[4] A. Fiat and A. Shamir, "How To Prove Yourself: Practical Solutions to Identification and Signature Problems", *Advances in Cryptology* – Crypto '86, A. M. Odlyzko (ed), Lecture Notes in Computer Science 263, pp. 186–194, Springer Verlag, 1987.
`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.8796`

[5] D. E. Knuth, *The Art of Computer Programming, Volume 2, Seminumerical Algorithms*, Addison-Wesley, Third edition, 1998. ISBN 0-201-89684-2.
`http://www.informit.com/title/0201896842`

[6] N. Koblitz, "Elliptic Curve Cryptosystems", *Mathematics of Computation* 48(177), pp. 203–209, January 1987.
`http://www.jstor.org/pss/2007884`

[7] N. Koblitz, *A Course in Number Theory and Cryptography*, Graduate Texts in Mathematics 114, Springer Verlag, Second edition, 1994. ISBN 978-0-387-94293-3.
`http://www.springer.com/math/numbers/book/978-0-387-94293-3`

[8] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", *Advances in Cryptology* – CRYPTO '96, N. Koblitz (ed), Lecture Notes in Computer Science 1109, pp. 104–113, Springer-Verlag, 1996.
`http://www.springerlink.com/content/4el17cvre3gxt4gd/`

[9] P. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis", *Advances in Cryptology* – CRYPTO '99, M. Wiener (ed), Lecture Notes in Computer Science 1666, pp. 388–397, Springer-Verlag, 1999.
`http://www.springerlink.com/content/kx35ub53vtrkh2nx/`

[10] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications*, Cambridge University Press, Second edition, 1994. ISBN 9780521460941.
`http://www.cambridgeuniversitypress.com/catalogue/catalogue.asp?isbn=9780521460941`

[11] G. L. Miller, "Riemann's Hypothesis and Tests for Primality", *J. Computer and System Sciences* 13(3), pp. 300–317, 1976.
`http://www.cs.cmu.edu/~glmiller/Publications/b2hd-Mi76.html`

[12] V. Miller, "Uses of Elliptic Curves in Cryptography", *Advances in Cryptology* – CRYPTO '85: PROCEEDINGS, H. C. Williams (ed), Lecture Notes in Computer Science 218, pp. 417–426, Springer Verlag, 1986.
`http://www.springerlink.com/content/4lfhkd08684v3wyl/`

[13] P. L. Montgomery, "Modular Multiplication Without Trial Division", *Mathematics of Computation* 44(170), pp. 519–521, April 1985.
`http://www.jstor.org/pss/2007970`

[14] J.-J. Quisquater and C. Couvreur, "Fast Decipherment Algorithm for RSA Public-key Cryptosystem", *Electronics Letters* 18(21), pp. 905–907, October 1982.
`http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4246955`

[15] J.-J. Quisquater and D. Samyde, "ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards", *Smart Card Programming and Security (e-Smart 2001)*, I. Attali & T. Jensen (eds), Lecture Notes in Computer Science 2140, pp. 200–210, Springer-Verlag, 2001.
`http://www.springerlink.com/content/chmydkq8x5tgdrce/`

[16] M. O. Rabin, "Probabilistic Algorithm for Testing Primality", *J. Number Theory* 12(1), pp. 128–138, February 1980.
`http://dx.doi.org/10.1016/0022-314X(80)90084-0`

[17] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM* 21(2), pp. 120–126, February 1978.
`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.5588`

[18] C. P. Schnorr, "Efficient Signature Generation by Smart Cards", *Journal of Cryptology* 4, pp. 161–174, 1991.
`http://www.springerlink.com/content/w037127811042441/`

[19] D. R. Stinson, *Cryptography: Theory and Practice*, CRC Press, Third edition, 2005. ISBN 9781584885085.
`http://www.crcpress.com/product/isbn/9781584885085`