MATHEMATICS IN CRYPTOGRAPHY

DR.S.VASUNDHARA

Abstract: Security is the most challenging aspects in the internet and network application. These days the applications like Internet and networks are growing very fast, thereby the importance and the value of the exchanged data over the internet or other media types are increasing. for secure communication the cryptography is essential. In information security, Cryptography algorithm is very important. Cryptography is subdivided into two - Symmetric and Asymmetric key cryptography. Cryptography has come up as a solution in information security system against various attacks. Thus this paper provides a self-contained treatment of mathematical cryptography with limited mathematical background. we suggest briefly reviewing the relevant concepts in number theory.

Keywords: Number theory, Cryptography

Introduction: The security of communications and commerce in a digital age relies on the modern incarnation of the ancient art of codes and ciphers. Underlying the birth of modern cryptography is a great deal of fascinating mathematics, some of which has been developed for cryptographic applications, but much of which is taken from the classical mathematical canon. The principal goal of this book is to introduce the reader to a variety of mathematical topics while simultaneously integrating the mathematics into a description of modern public key cryptography. For thousands of years, all codes and ciphers relied on the assumption that the people attempting to communicate call them Bob and Alice, share a secret key that their adversary, call her Eve, does not possess. Bob uses the secret key to encrypt his message, Alice uses the same secret key to decrypt the message, and poor Eve, not knowing the secret key, is unable to perform the decryption. A disadvantage of these private key cryptosystems is that Bob and Alice need to exchange the secret key before they can get started. During the 1970s, the astounding idea of public key cryptography burst upon the scene.1 In a public key cryptosystem, Alice has two keys, a public encryption key KPub and a private (secret) decryption key KPri. Alice publishes her public key KPub, and then Adam and Bob and Carl and everyone else can use KPub to encrypt messages and send them to Alice. The idea underlying public key cryptography is that although everyone in the world knows KPub and can use it to encrypt messages, only Alice, who knows the private key KPri, is able to decrypt messages.

The advantages of a public key cryptosystem are manifold. For example, Bob can send Alice an encrypted message even if they have never previously been in direct contact. But although public key cryptography is a fascinating theoretical concept, it is not at all clear how one might create a public key cryptosystem. It turns out that public key cryptosystems can be based on hard mathematical problems. More precisely, one looks for a mathematical problem that is initially hard to solve, but that becomes easy to solve if one knows some extra piece of information. of course, private key cryptosystems have not disappeared. Indeed, they are more important than ever, since they tend to be significantly more efficient than public key cryptosystems. Thus in practice, if Bob wants to send Alice a long message, he first uses a public key cryptosystem to send Alice the key for a private key cryptosystem, and then he uses the private key cryptosystem to encrypt his message. The most efficient modern private key cryptosystems, such as DES and AES, rely for their security on repeated application of various mixing operations that are hard to un mix without the private key. Thus although the subject of private key cryptography is of both theoretical and practical importance, the connection with fundamental underlying mathematical ideas is much less pronounced than it is with public key cryptosystems. For that reason, especially public key cryptosystems and digital signatures. Modern mathematical cryptography draws on many areas of mathematics, including especially number theory, abstract algebra (groups, rings, fields), probability, statistics, and information theory, so the prerequisites for studying the subject can seem formidable. so we take the time to introduce each required mathematical topic in sufficient depth as it is needed.

Encryption of a message means the information in it is hidden so that anyone who's reading (or listening to) the message, can't understand any of it unless he/she can break the encryption. An original plain message is called plaintext and an encrypted one crypto text. When encrypting you need to have a so-called key, a usually quite complicated parameter that you can use to change the encryption. If the encrypting procedure remains unchanged for a long time, the probability of breaking the encryption will in practice increase substantially. Naturally different users need to have their own keys, too.The receiver of the message decrypts it, for which he/she needs to

have his/her own key. Both the encrypting key and decrypting key are very valuable for an eavesdropper, using the encrypting key he/she can send encrypted fake messages and using the decrypting key he/she can decrypt messages not meant to him/her. In symmetric cryptosystems both the encrypting key and the decrypting key are usually the same.

An encryption procedure is symmetric, if the encrypting and decrypting keys are the same or it's easy to derive one from the other. In non symmetric encryption the decrypting key can't be derived from the encrypting key with any small amount of work. In that case the encrypting key can be public while the decrypting key stays classified. This kind of encryption procedure is known as public-key cryptography, correspondingly symmetric encrypting is called secret key cryptography. The problem with symmetric encrypting is the secret key distribution to all parties, as keys must also be updated every now and then. Symmetric encryption can be characterized as a so called cryptosystem which is an ordered quintet (P, C, K, E, D), where • P is the finite message space (plaintexts). • C is the finite crypto text space (crypto texts). • K is the finite key space. • for every key k ∈ K there is an encrypting function ek ∈ E and a decrypting function dk ∈ D. E is called the encrypting function space which includes every possible encrypting function and D is called the decrypting function space which includes every possible decrypting function.

The creation of public key cryptography by Diffie and Hellman in 1976 and the subsequent invention of the RSA public key cryptosystem by Rivest, Shamir, and Adleman in 1978 are watershed events in the long history of secret communications. It is hard to overestimate the importance of public key cryptosystems and their associated digital signature schemes in the modern world of computers and the Internet. Public key cryptography draws on many areas of mathematics, including number theory, abstract algebra, probability, and information theory.

Much of modern cryptography is built on the foundations of algebra and number theory. So before we explore the subject of cryptography, we need to develop some important tools. At the most basic level, Number Theory is the study of the natural numbers 1, 2, 3, 4, 5, 6,..., or slightly more generally, the study of the integers ..., −5, −4, −3, −2, −1, 0, 1, 2, 3, 4, 5,.... The set of integers is denoted by the symbol Z. Integers can be added, subtracted, and multiplied in the usual way, and they satisfy all the usual rules of arithmetic (commutative law, associative law, distributive law, etc.). The set of integers with their addition and multiplication rules are an example of a ring.

If a and b are integers, then we can add them a + b, subtract them a – b, and multiply them a · b. In each case, we get an integer as the result. This property of staying inside of our original set after applying operations to a pair of elements is characteristic of a ring. But if we want to stay within the integers, then we are not always able to divide one integer by another. For example, we cannot divide 3 by 2, since there is no integer that is equal to 3divides 2 . This leads to the fundamental concept of divisibility.

Definition 1.1. Let a and b be integers with b = 0. We say that b divides a, or that a is divisible by b, if there is an integer c such that a = bc. We write b | a to indicate that b divides a. If b does not divide a, then we write b a.

1.2. Divisibility and greatest common divisors: We have 847 | 485331, since 485331 = 847 · 573. On the other hand, 355 259943, since when we try to divide 259943 by 355, we get a remainder of 83. More precisely, 259943 = 355 · 732 + 83, so 259943 is not an exact multiple of 355.

Remark 1.3: Notice that every integer is divisible by 1. The integers that are divisible by 2 are the even integers, and the integers that are not divisible by 2 are the odd integers. There are a number of elementary divisibility properties, some of which we list in the following proposition.

Proposition 1.4: Let a,b,c ∈ Z be integers. (a) If a | b and b | c, then a | c. (b) If a | b and b | a, then a = ±b. (c) If a | b and a | c, then a | (b + c) and a | (b – c).

1.5. Definition. A common divisor of two integers a and b is a positive integer d that divides both of them. The greatest common divisor of a and b is, as its name suggests, the largest positive integer d such that d | a and d | b. The greatest common divisor of a and b is denoted gcd(a,b). If there is no possibility of confusion, it is also sometimes denoted by (a,b). (If a and b are both 0, then gcd(a,b) is not defined.) It is a curious fact that a concept as simple as the greatest common divisor has many applications. We'll soon see that there is a fast and efficient method to compute the greatest common divisor of any two integers, a fact that has powerful and far-reaching consequences.

Example 1.5: The greatest common divisor of 12 and 18 is 6, since 6 | 12 and 6 | 18 and there is no larger number with this property. Similarly, gcd(748, 2024) = 44. One way to check that this is correct is to make lists of all of the positive divisors of 748 and of 2024. Divisors of 748 = {1, 2, 4, 11, 17, 22, 34, 44, 68, 187, 374, 748}, Divisors of 2024 = {1, 2, 4, 8, 11, 22, 23, 44, 46, 88, 92, 184, 253, 506, 1012, 2024}. Examining the two lists, we see that the largest common entry is 44. Even from this small example, it is clear that this is not a very efficient method. If we ever need to compute greatest common divisors of large numbers, we will have to find a more efficient approach.

The key to an efficient algorithm for computing greatest common divisors is division with remainder,

which is simply the method of "long division" that you learned in elementary school. Thus if a and b are positive integers and if you attempt to divide a by b, you will get a quotient q and a remainder r, where the remainder r is smaller than b. For example,

13 R 9 17) 230 17 60 51 9 so 230 divided by 17 gives a quotient of 13 with a remainder of 9. What does this last statement really mean? It means that 230 can be written as 230 = 17 · 13 + 9, where the remainder 9 is strictly smaller than the divisor 17.

Definition: (Division Algorithm) Let a and b be positive integers. Then a divided by b has quotient q and remainder r means that a = b · q + r with 0 ≤ r< r. Then the same reasoning shows that

gcd(b,r) = gcd(r,r). Continuing this process, the remainders become smaller and smaller, until eventually we get a remainder of 0, at which point the final value gcd(s, 0) = s is equal to the gcd of a and b. We illustrate with an example and then describe the general method, which goes by the name Euclidean algorithm. 1.2. Divisibility and greatest common divisors Example 1.6. We compute gcd(2024, 748) using the Euclidean algorithm, which is nothing more than repeated division with remainder. Notice how the quotient and remainder on each line become the new a and b on the subsequent line: 2024 = 748 · 2 + 528 748 = 528 · 1 + 220 528 = 220 · 2 + 88 220 = 88 · 2 + 44 ← gcd = 44 88 = 44 · 2+ 0

Theorem 1.6 (The Euclidean Algorithm):

Let a and b be positive integers with a ≥ b. The following algorithm computes gcd(a,b) in a finite number of steps.

(1) Let r0 = a and r1 = b.

(2) Set i = 1.

(3) Divide ri−1 by ri to get a quotient qi and remainder ri+1, ri−1 = ri · qi + ri+1 with 0 ≤ ri+1 < ri.

(4) If the remainder ri+1 = 0, then ri = gcd(a,b) and the algorithm terminates.

(5) Otherwise, ri+1 > 0, so set i = i + 1 and go to Step 3. The division step (Step 3) is executed at most 2log2(b)+1 times.

Proof. The Euclidean algorithm consists of a sequence of divisions with remainder as illustrated in Figure 1.2 (remember that we set r0 = a and r1 = b). a = b · q1 + r2 with 0 ≤ r2 < b, b = r2 · q2 + r3 with 0 ≤ r3 < r2, r2 = r3 · q3 + r4 with 0 ≤ r4 < r3, r3 = r4 · q4 + r5 with 0 ≤ r5 < r4, ... rt−2 = rt−1 · qt−1 + rt with 0 ≤ rt < rt−1, rt−1 = rt · qt Then rt = gcd(a,b). Figure 1.2: The Euclidean algorithm step by step The ri values are strictly decreasing, and as soon as they reach zero the algorithm terminates, which proves that the algorititthm does finish in a finite Number of steps. Further, at each iteration of Step 3 we have an equation of the form ri−1 = ri · qi + ri+1. This equation implies that any common divisor of ri−1 and ri is also a divisor of ri+1, and similarly it implies that any common divisor of ri and ri+1 is also a divisor of ri−1. Hence gcd(ri−1,ri) = gcd(ri,ri+1) for all i = 1, 2, 3,... . (1.2) However, as noted above, we eventually get to an ri that is zero, say rt+1 = 0. Then rt−1 = rt · qt, so gcd(rt−1,rt) = gcd(rt · qt,rt) = rt. But equation (1.2) says that this is equal to gcd(r0,r1), i.e., to gcd(a,b), which completes the proof that the last nonzero remainder in the Euclidean algorithm is equal to the greatest common divisor of a and b. It remains to estimate the efficiency of the algorithm. We noted above that since the ri values are strictly decreasing, the algorithm terminates, and indeed since r1 = b, it certainly terminates in at most b steps. However, this upper bound is far from the truth. We claim that after every two iterations of Step 3, the value of ri is at least cut in half. In other words: Claim: ri+2 < 1 2 ri for all i = 0, 1, 2,... . We prove the claim by considering two cases. Case I: ri+1 ≤ 1 2 ri We know that the ri values are strictly decreasing, so ri+2 < ri+1 ≤ 1 2 ri. Case II: ri+1 > 1 2 ri Consider what happens when we divide ri by ri+1. The value of ri+1 is so large that we get ri = ri+1 · 1 + ri+2 with ri+2 = ri − ri+1 < ri − 1 2 ri = 1 2 ri. We have now proven our claim that ri+2 < 1 2 ri for all i. Using this inequality repeatedly, we find that r2k+1 < 1 2 r2k−1 < 1 4 r2k−3 < 1 8 r2k−5 < 1 16 r2k−7 < ··· < 1 2k r1 = 1 2k b. Hence if 2k ≥ b, then r2k+1 < 1, which forces r2k+1 to equal 0 and the algorithm to terminate. In terms of Figure 1.2, the value of rt+1 is 0, so we 1.2. Divisibility and greatest common divisors have t + 1 ≤ 2k + 1, and thus t ≤ 2k. Further, there are exactly t divisions performed in Figure 1.2, so the Euclidean algorithm terminates in at most 2k iterations. Choose the smallest such k, so 2k ≥ b > 2k−1. Then # of iterations ≤ 2k = 2(k − 1) + 2 < 2log2(b)+2, which completes the proof of Theorem 1.7. Remark : We proved that the Euclidean algorithm applied to a and b with a ≥ b requires no more than 2log2(b) + 1 iterations to compute gcd(a,b). This estimate can be somewhat improved. It has been proven that the Euclidean algorithm takes no more than 1.45log2(b)+1.68 iterations, and that the average number of iterations for randomly chosen a and b is approximately 0.85log2(b)+0.14.

Remark 1.8: One way to compute quotients and remainders is by long division, You can speed up the process using a simple calculator. The first step is to divide a by b on your calculator, which will give a real number. Throw away the part after the decimal point to get the quotient q. Then the remainder r can be computed as r = a − b · q. For example, let a = 2387187 and b = 27573. Then a/b ≈ 86.57697748, so q = 86 and r = a − b · q = 2387187 − 27573 · 86 = 15909. If you need just the remainder, you can instead take the decimal part (also sometimes called the fractional part) of a/b and multiply it by b. Continuing with our example, the decimal part of a/b ≈ 86.57697748 is 0.57697748, and multiplying by b = 27573 gives 27573 · 0.57697748 = 15909.00005604. Rounding this off gives r = 15909.

After performing the Euclidean algorithm on two numbers, we can work our way back up the process to obtain an extremely interesting formula. Before giving the general result, we illustrate with an example.

Example 1.9: Recall that in Example 1.6 we used the Euclidean algorithm to compute gcd(2024, 748) as follows: 2024 = 748 · 2 + 528 748 = 528 · 1 + 220 528 = 220 · 2 + 88 220 = 88 · 2 + 44 ← gcd = 44 88 = 44 · 2+ 0 16 1. We let a = 2024 and b = 748, so the first line says that 528 = a – 2b.

We substitute this into the second line to get b = (a – 2b) · 1 + 220, so 220 = –a + 3b. We next substitute the expressions 528 = a – 2b and 220 = –a + 3b into the third line to get a – 2b = (–a + 3b) · 2 + 88, so 88 = 3a – 8b. Finally, we substitute the expressions 220 = –a + 3b and 88 = 3a – 8b into the penultimate line to get –a + 3b = (3a – 8b) · 2 + 44, so 44 = –7a + 19b. In other words, –7 · 2024 + 19 · 748 = 44 = gcd(2024, 748), so we have found a way to write gcd(a,b) as a linear combination of a and b using integer coefficients. In general, it is always possible to write gcd(a,b) as an integer linear combination of a and b, a simple sounding result with many important consequences.

Theorem 1.10 (Extended Euclidean Algorithm): Let a and b be positive integers. Then the equation au + bv = gcd(a,b) always has a solution in integers u and v. for an efficient algorithm to find a solution.) If (uo,vo) is any one solution, then every solution has the form u = uo + b · k gcd(a,b) and v = vo – a · k gcd(a,b) for some k ∈ Z.

Proof. solve the first line for r2 = a – b · q1 and substitute it into the second line to get b = (a – b · q1) · q2 + r3, so r3 = –a · q2 + b · (1 + q1q2). Next substitute the expressions for r2 and r3 into the third line to get a – b · q1 = –a · q2 + b · (1 + q1q2) q3 + r4. 1.2. Divisibility and greatest common divisors After rearranging the terms, this gives r4 = a · (1 + q2q3) – b · (q1 + q3 + q1q2q3). The key point is that r4 = a · u + b · v, where u and v are integers. It does not matter that the expressions for u and v in terms of q1,q2,q3 are rather messy. Continuing in this fashion, at each stage we find that ri is the sum of an integer multiple of a and an integer multiple of b. Eventually, we get to rt = a · u+b · v for some integers u and v. But rt = gcd(a,b), which completes the proof of the first part of the theorem. We leave the second part as especially important case of the extended Euclidean algorithm arises when the greatest common divisor of a and b is 1. In this case we give a and b a special name.

Definition: Let a and b be integers. We say that a and b are relatively prime if gcd(a,b) = 1.

More generally, any equation Au + Bv = gcd(A,B) can be reduced to the case of relatively prime numbers by dividing both sides by gcd(A,B). Thus A gcd(A,B) u + B gcd(A,B) v = 1,

2.1 Modular arithmetic: Definition. Let m ≥ 1 be an integer. We say that the integers a and b are congruent modulo m if their difference a – b is divisible by m. We write a ≡ b (mod m) to indicate that a and b are congruent modulo m. The number m is called the modulus. Our clock examples may be written as congruences using the modulus m = 12: 6 + 9 = 15 ≡ 3 (mod 12) and 2 – 3 = –1 ≡ 11 (mod 12). Example 1.12. We have 17 ≡ 7 (mod 5), since 5 divides 10 = 17 – 7. On the other hand, 19 ≡ 6 (mod 11), since 11 does not divide 13 = 19 – 6. Notice that the numbers satisfying a ≡ 0 (mod m) are the numbers that are divisible by m, i.e., the multiples of m. The reason that congruence notation is so useful is that congruences behave much like equalities, as the following proposition indicates.

Proposition 2.12: Let m ≥ 1 be an integer. (a) If a1 ≡ a2 (mod m) and b1 ≡ b2 (mod m), then a1 ± b1 ≡ a2 ± b2 (mod m) and a1 · b1 ≡ a2 · b2 (mod m). (b) Let a be an integer. Then a · b ≡ 1 (mod m) for some integer b if and only if gcd(a,m)=1. If such an integer b exists, then we say that b is the (multiplicative) inverse of a modulo m. (We say "the" inverse, rather than "an" inverse, because any two inverses are congruent modulo m.)

Suppose first that gcd(a,m) = 1. Then Theorem 2.11 tells us that we can find integers u and v satisfying au + mv = 1. This means that au – 1 = –mv is divisible by m, so by definition, au ≡ 1 (mod m). In other words, we can take b = u. For the other direction, suppose that a has an inverse modulo m, say a · b ≡ 1 (mod m). This means that ab – 1 = cm for some integer c. It follows that gcd(a,m) divides ab – cm = 1, so gcd(a,m) = 1. This completes the proof that a has an inverse modulo m if and only if gcd(a,m) = 1.

Proposition 2.13 says that if gcd(a,m) = 1, then there exists an inverse b of a modulo m. This has the curious consequence that the fraction b–1 = 1/b then has a meaningful interpretation in the world of integers modulo m.

Modular arithmetic Example 2.14. We take m = 5 and a = 2. Clearly gcd(2, 5) = 1, so there exists an inverse to 2 modulo 5. The inverse of 2 modulo 5 is 3, since 2·3 ≡ 1 (mod 5), so 2–1 ≡ 3 (mod 5). Similarly gcd(4, 15) = 1 so 4–1 exists modulo 15. In fact 4 · 4 ≡ 1 (mod 15) so 4 is its own inverse modulo 15. We can even work with fractions a/d modulo m as long as the denominator is relatively prime to m.

For example, we can compute 5/7 modulo 11 by first observing that 7 · 8 ≡ 1 (mod 11), so 7–1 ≡ 8 (mod 11). Then 5 7 = 5 · 7–1 ≡ 5 · 8 ≡ 40 ≡ 7 (mod 11).

Remark 2.15. In the preceding examples it was easy to find inverses modulo m by trial and error. However, when m is large, it is more challenging to compute a–1 modulo m. Note that we showed that inverses exist by using the extended Euclidean algorithm (Theorem 1.10). In order to actually compute the u

and v that appear in the equation au + mv = gcd(a,m), we can apply the Euclidean algorithm directly as we did in Example 1.9, or we can use the algorithm given in Exercise 1.11. In any case, since the Euclidean algorithm takes only 2log2(b) + 3 iterations to compute gcd(a,b), it takes only a small multiple of log2(m) steps to compute a–1 modulo m. We now continue our development of the theory of modular arithmetic.

Conclusion: A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

Appendix: Appendixes, if needed, appear before the acknowledgment.

Acknowledgment: The preferred spelling of the word "acknowledgment" in American English is without an "e" after the "g." Use the singular heading even if you have many acknowledgments. Avoid expressions such as "One of us (S.B.A.) would like to thank ... ." Instead, write "F. A. Author thanks ... ." Sponsor and financial support

References:

1. KOBLITZ, N.: A Course in Number Theory and Cryptography. Springer–Verlag (2001)
2. KOBLITZ, N.: Algebraic Aspects of Cryptography. Springer–Verlag (2004)
3. KONHEIM, A.G.: Cryptography. A Primer. Wiley (1981)
4. KRANAKIS, E.: Primality and Cryptography. Wiley (1991)
5. LIDL, R. & NIEDERREITER, H.: Finite Fields. Cambridge University Press (2008)
6. LIPSON, J.D.: Elements of Algebra and Algebraic Computing. Addison–Wesley (1981)
7. MAO, W.: Modern Cryptography. Theory and Practice. Pearson Education (2004)
8. MCELIECE, R.J.: Finite Fields for Computer Scientists and Engineers. Kluwer (1987)
9. MENEZES, A. & VAN OORSCHOT, P. & VANSTONE, S.: Handbook of Applied Cryptography.
10. CRC Press (2001)
11. MIGNOTTE, M.: Mathematics for Computer Algebra. Springer–Verlag (1991)
12. MOLLIN, R.A.: An Introduction to Cryptography. Chapman & Hall / CRC (2006)
13. MOLLIN, R.A.: RSA and Public-Key Cryptography. Chapman & Hall / CRC (2003)
14. MOLLIN, R.A.: Codes. The Guide to Secrecy from Ancient to Modern Times. Chapman
15. & Hall / CRC (2005)
16. NIELSEN, M.A. & CHUANG, I.L.: Quantum Computation and Quantum Information.
17. Cambridge University Press (2000)
18. PAAR, C. & PELZL, J.: Understanding Cryptography. A Textbook for Students and
19. Practitioners. Springer–Verlag (2009)
20. RIESEL, H.: Prime Numbers and Computer Methods for Factorization. Birkhäuser
21. (1994)
22. ROSEN, K.H..: Elementary Number Theory. Longman (2010)
23. ROSING, M.: *Implementing Elliptic Curve Cryptography*. Manning Publications (1998)
24. SALOMAA, A.: *Public-Key Cryptography*. Springer–Verlag (1998)

* * *

Dr.S.Vasundhara
Asst. Professor of Mathematics
G. Narayanamma Institute of Technology &Science (women)
Shaikpet, Hyderabad 500104