An Oblivious Transfer Scheme
in Gaussian Arithmetic

Kenneth K. Nwabueze

Department of Mathematics,
University of Brunei Darussalam,
BE 1410, Gadong, BRUNEI

Abstract

Most oblivious transfer schemes blend the capabilities of modern high-speed computers with Gaussian arithmetic. In this expository note, the basic mathematics used in a scheme that converts a remote call for “heads” or “tails” into the problem of factoring a large number is discussed.

Mathematics Subject Classification: 94A05, 11A99

Keywords: Gaussian arithmetic, primes, remote coin tosses and oblivious transfer.

1 Introduction

In this era of electronic mails and super computers, various methods are used to guarantee that transactions done through electronic channels are correctly executed. For example, there is need to ensure that electronic messages arrive intact, that funds are electronically transferred correctly, and that contracts and agreements are properly executed over the internet. Without adequate security electronic transactions are not safe. It is amazing that the protocol for information security of some of the most sophisticated electronic transmitted transactions employ only the fundamental mathematical ideas that were discovered many centuries ago. At the heart of most security protocols is a procedure called the “oblivious transfer”. The oblivious transfer depends on two crucial simple mathematical ideas: first, knowing quick ways to determine whether or not a large number is a prime (microcomputers can do this in seconds); second, the belief that a number which is a product two roughly equal primes can take years to factor, even with the fastest available computers (there
is convincing mathematical evidence to support this). The security of oblivious transfer schemes hinges on the difficulty of factoring large numbers. The oblivious transfer schemes make appearances in protocols, such as, in sending certified mails, electronic signatures, in bank automated transaction systems, in remote coin tosses, etcetera. In this note, we present an expository gist of some basic simple mathematics, rooted in ancient Greece, which are utilized in enhancing security in our recent information society. As an example, we discuss the protocol of remote coin toss.

2 Basic notes

In this section we survey the rudiments of elementary algebra and number theory required for the implementation of some oblivious transfer schemes.

Gaussian Arithmetic

Gaussian arithmetic, also called modular arithmetic, is a handy key for locking up secrets in some oblivious transfer schemes. Let \( p \) denote a prime number. By \((\mod p)\) we mean arithmetic done modulo \( p \); that is, divide by \( p \), and keep the remainder \( r \), where \( 0 < r < p \). We shall use the notation \( a \equiv b (\mod n) \) (read as \( a \) is congruent to \( b \) modulo \( n \)) for the expression: \( n \) divides \( a \), with a left over \( b \). The number \( n \) is called the “modulus”. Note that \( a \equiv b (\mod n) \) also implies that \( a (\mod n) \equiv b (\mod n) \). Therefore, there exist some integers \( k_1 \) and \( k_2 \), such that \( a = k_1 \cdot n + r \), \( b = k_2 \cdot n + r \), \( (0 < r < n) \). This means \( a - b = (k_1 - k_2) \cdot n \); that is, \( n \) divides \( a - b \), written mathematically as \( n | (a - b) \).

The Group \( \mathbb{Z}_p^* \)

The set \( \mathbb{Z}_p^* := \{1, 2, \ldots, p-1\} \) is very important in the implementation of oblivious transfer schemes because multiplication and exponentiation take place within this set. The following facts about this set can easily be verified:

- if we multiply any two numbers \((\mod p)\), the result is a number in the set; meaning that the set is closed under multiplication.
- for any number \( n \), there exists another number \( n^{-1} \), such that \( n \cdot n^{-1} = 1 (\mod p) \); meaning that any number in the set has a multiplicative inverse.

The above two properties imply that the set \( \mathbb{Z}_p^* \) is a group under multiplication \((\mod p)\). We also have that \( \mathbb{Z}_p^* \) is a group under exponentiation, because the \( k \)th power of a number is simply the multiplication of that number by itself \( k \) times. Note that exponentiating a number in the set yields another number in the set. Also note that because \( p \) is a prime, each element has a multiplicative inverse in \( \mathbb{Z}_p^* \). Moreover, the greatest common divisor \((gcd)\) of \( p \) and any number in this set is 1. The following remarks are worthwhile:

- \( \mathbb{Z}_p \neq \mathbb{Z}_p^* \); in \( \mathbb{Z}_p^* \), we omit 0, because 0 does not have a multiplicative inverse,
and if we add 0 to the set \( \mathbb{Z}_p^* \), we get the set \( \mathbb{Z}_p \), which consists precisely of 0 and all remainders \((\text{mod } p)\).

- the modulo arithmetic considered above for \( p \) is not the same for composite numbers. It is an easy exercise to verify that if we perform multiplication in \((\text{mod } n)\) (for \( n \) composite), then some elements may have no inverse in the set.

An element \( a \) in \( \mathbb{Z}_p^* \) is said to be a generator \((\text{mod } p)\) if the set consists of elements made up of powers of \( a \), that is, the set \( \{a^1(\text{mod } p), a^2(\text{mod } p), \ldots, a^{(p-1)}(\text{mod } p)\} \) contains in some order all the members of \( \mathbb{Z}_p^* \). This means that the set \( \mathbb{Z}_p^* \) represents a rearrangement of \( \{a, a^2, \ldots, a^{(p-1)}\} \) after all calculations are done \((\text{mod } p)\). In general, a generator-tuple \((\text{mod } p)\) is a set of different \( k \) generators. This means \( \{a_1, \ldots, a_k\} \) is a generator-tuple if each \( a_i \) is a generator \((\text{mod } p)\), where \( a_i \) is not equal to \( a_j \) if \( i \) is not equal to \( j \). Note that sometimes an element may generate only a subgroup of a group. A group generated by an element \( a \) is said to have order \( q(\text{mod } p) \) if \( q \) is the smallest power such that \( a^q \equiv 1(\text{mod } p) \). For a prime \( q \) such that \( 1 < q < p \), we define \( G(q) \) as the group (or subgroup) of prime order \( q(\text{mod } p) \), if for some generator \( a \ (1 < a < p) \), we have that the set \( \{a, a^2, a^3, \ldots, a^q\} \) is a subset of \( \mathbb{Z}_p^* \). This means that the powers of \( a \) yield each of the elements in the subgroup. By definition \( q \) is the smallest power of \( a \) that gives \( 1(\text{mod } p) \); therefore \( a^q \equiv 1(\text{mod } p) \). So powers larger than \( q \) simply start over and run through the same set of numbers. If \( a^q \equiv 1(\text{mod } p) \), then \( a^{(q+1)} \equiv a(\text{mod } p), a^{(q+2)} \equiv a^2(\text{mod } p) \), etcetera. Finally, note that if \( a \) is an element of the group \( \mathbb{Z}_p^* \), then \( a \) is a generator of \( \mathbb{Z}_p^* \) if \( a \) is an element of order \( p-1 \). This means, if \( a^{p-1} \equiv 1(\text{mod } p) \), and no lower power gives \( 1(\text{mod } p) \) because it necessarily follows that the powers of \( a(\text{mod } p) \), namely, \( a^1, a^2, \ldots, a^{p-1} \), runs through all the numbers \( 1, 2, \ldots, p-1 \).

**Fermat's Little Theorem**

Fermat's theorem tells us that for any prime \( p \), and number \( n \) not divisible by \( p \), we have \( n^{p-1} \equiv 1(\text{mod } p) \). \( p \) does not divide the integers \( 1, 2, 3, \ldots, p-1 \), and by Fermat’s theorem, any of these integers raised to the power \( p-1 \) equals \( 1(\text{mod } p) \). So for an element \( n \) of \( \mathbb{Z}_p^* \), we have \( n^{p-1} \equiv 1(\text{mod } p) \). As a consequence of Fermat's theorem, we have that the order \( q \) of any element of a multiplicative group \((\text{mod } p)\) must divide \( p-1 \). To see this, recall that if an element \( a \) is of order \( q(\text{mod } p) \), then \( a^q \equiv 1(\text{mod } p) \). However, it also follows by Fermat’s theorem that \( a^{p-1} \equiv 1(\text{mod } p) \). Therefore if \( q \) does not divide \( p-1 \) then there exist some number \( n \), such that \( p-1 = n \cdot q + r \), for \( 0 < r < q \). Thus we would have that \( 1 \equiv a^{p-1} \equiv a^{kq+r} \equiv (a^q)^k \cdot a^r \equiv 1 \cdot a^r(\text{mod } p) \). This implies \( a^r = 1 \); which in turn implies that \( a \) has order \( r(\text{mod } p) \) which is less than \( q \). This is a contradiction, in view of the minimality of \( q \).

**Wilson's theorem**
Wilson’s theorem gives necessary and sufficient condition for primality testing, namely an integer \( p > 1 \) is prime if and only if \( (p - 1)! \equiv -1 (\text{mod } p) \).

**Euler’s function and theorem**

Euler’s function for a positive integer \( n \), denoted by \( \Phi(n) \), is the set of all numbers less than \( n \) that are relatively prime to \( n \). That is, the number of positive integers \( k \) (\( 0 < k < n \)) with \( \gcd(k, n) = 1 \). For a prime \( p \) we have that all positive numbers less than \( p \) are relatively prime to \( p \), and so \( \Phi(p) = p - 1 \).

It can be shown that for a given number \( n \) and any number \( k \) relatively prime to \( n \), the relation \( k^{\Phi(n)} \equiv 1 (\text{mod } n) \) holds. This is the statement of the Euler’s theorem. Note that this theorem applies to any composite numbers \( n \), as well as prime numbers. We can say outright what the number of generators (\( \text{mod } p \)) are, using a result which states that for a prime \( p \), the number of generators (\( \text{mod } p \)) is \( \Phi(p - 1) \). Now take the group \( G(q) \) of prime order \( q (\text{mod } p) \), where both \( p \) and \( q \) are distinct primes. Since the order of any element (\( \text{mod } p \)) must divide \( p - 1 \), it follows that \( q \) must divide \( p - 1 \). The number of generators of \( G(q) \) are precisely \( \Phi(m) \) for which \( m \) divides \( p - 1 \). This means that there are \( q - 1 \) generators of the subgroup \( G(q) \) of prime order \( q (\text{mod } p) \). This fact guarantees that for large \( q \) we have a lot of generators to choose from.

**Chinese Remainder Theorem**

It is well known that if \( p \) and \( q \) are relatively prime, then the congruences \( x \equiv a_p (\text{mod } p) \), and \( x \equiv a_q (\text{mod } q) \) have a unique solution modulo \( m = pq \). This is the so-called Chinese remainder theorem. To check that this theorem is true; map the \( m \) numbers of \( \mathbb{Z}_m \) to their residues (\( \text{mod } p \)) and (\( \text{mod } q \)), that is, define \( f : \mathbb{Z}_m \rightarrow \mathbb{Z}_p \times \mathbb{Z}_q \) by \( x \rightarrow (x (\text{mod } p), x (\text{mod } q)) \). The map \( f \) is injective because \( f(x) = f(y) \) implies \( x = y \) in \( \mathbb{Z}_m \), and surjective because \( |\mathbb{Z}_m| = |\mathbb{Z}_p \times \mathbb{Z}_q| \). Indeed for every \( (a_p, a_q) \) in \( \mathbb{Z}_p \times \mathbb{Z}_q \) there exists a unique \( x \) in \( \mathbb{Z}_m \) with \( f(x) = (a_p, a_q) \).

**Quadratic Residues and Non-Quadratic Residues**

If \( m \) is a positive integer, we say that the integer \( a \) is a quadratic residue of \( m \) if \( \gcd(a, m) = 1 \) and the congruence \( x^2 \equiv a (\text{mod } m) \) has a solution. If the congruence \( x^2 \equiv a (\text{mod } m) \) has no solution, we say that \( a \) is a non-quadratic residue of \( m \). It is easy to show that if \( p \) is an odd prime and \( a \) is an integer not divisible by \( p \), then, the congruence \( x^2 \equiv a (\text{mod } p) \) has either no solutions or exactly two incongruent solutions modulo \( p \). It is also easy to check that if \( p \) is an odd prime, then there are exactly \( (p - 1)/2 \) quadratic residues of \( p \) and \((p - 1)/2 \) non-quadratic residues of \( p \) among the integers \( 1, 2, \ldots, p - 1 \). Let \( p \) be an odd prime and \( a \) an integer not divisible by \( p \). The Legendre symbol \( \left( \frac{a}{p} \right) \) is defined by \( \left( \frac{a}{p} \right) = 1 \) or \(-1 \) according as \( a \) is a quadratic residue of \( p \) or a non-quadratic residue of \( p \). Euler’s criterion says that if \( p \) is an odd prime and \( a \) is a positive integer not divisible by \( p \) then \( \left( \frac{a}{p} \right) \equiv a^{(p-1)/2} (\text{mod } p) \). For
more detailed explanation of all the excerpts in this section we refer the reader to [1], and [2].

3 Remote coin toss algorithm

One ingenious scheme for remote coin tosses blends the capabilities of modern, high-speed computers with elementary mathematics. In essence, this particular algorithm converts a call for “heads” or “tails” into the problem of factoring a large number. To win, the caller must find the two smaller numbers that the coin flipper multiplied to generate a large number. Before providing the generalized form of the algorithm for remote coin toss, we first illustrate its protocol using two prime numbers 7 and 13. We have chosen small prime numbers for ease of computation and illustrative purposes. In actual fact, very large numbers more than 120 digits are used so that the chances of anyone being able to factorize the numbers immediately are incredibly very remote.

Basic illustration

Suppose two parties A and B live in two different countries and can only communicate by e-mail or telephone. They want to toss a coin immune to cheating. They start the coin toss using their linked personal computers by one party, say A, choosing two prime numbers, say 7 and 13, and sends the product $7 \cdot 13 = 91$ to B. The objective here is for B to factorize the number 91 which is the product of two primes. If B factorizes 91, he wins the toss. A wins if he fails to do so. Below is the protocol.

Protocol

Step 1: A chooses two prime numbers, say 7 and 13, and multiplies the two numbers together to get $7 \cdot 13 = 91$. A sends the number 91 to B and keeps the numbers 7 and 13 secret.

Step 2: B checks whether A has sent him a number which cannot be factorized (thus cheating) by performing a primality test (for instance using Wilson’s theorem) on that number. If all is right, B randomly chooses a number between 1 and 91, for example 20. He squares 20 to get 400 and calculates the least remainder of 400 (mod 91) which is 36. He sends 36 to A and keeps 20 secret.

Step 3: A has to find numbers between 1 and 91 that generate the remainder 36 when squared and divided by 91, that is, solving the congruence $x^2 \equiv 36 \pmod{91}$. A knows that 91=7⋅13; obtains the congruences:

1. $x^2 \equiv 36 \pmod{7} \equiv 1 \pmod{7}$;
2. $x^2 \equiv 36 \pmod{13} \equiv 10 \pmod{13}$.

We know that if a quadratic congruence is solvable, it has two solutions. Solving congruence (1) gives $x \equiv 1$ or $6 \pmod{7}$, and solving congruence (2) gives $x \equiv 6$ or $7 \pmod{13}$. Therefore, we have the following four sets of simultaneous congruences

\[
\begin{align*}
(i) & \quad \left\{ \begin{array}{l}
x \equiv 1 \pmod{7} \\
x \equiv 6 \pmod{13}
\end{array} \right. \\
(ii) & \quad \left\{ \begin{array}{l}
x \equiv 1 \pmod{7} \\
x \equiv 7 \pmod{13}
\end{array} \right. \\
(iii) & \quad \left\{ \begin{array}{l}
x \equiv 6 \pmod{7} \\
x \equiv 6 \pmod{13}
\end{array} \right. \\
(iv) & \quad \left\{ \begin{array}{l}
x \equiv 6 \pmod{7} \\
x \equiv 7 \pmod{13}
\end{array} \right.
\end{align*}
\]
(iv) \( \begin{cases} x \equiv 6 \pmod{7} \\ x \equiv 7 \pmod{13} \end{cases} \). Solving the systems (i), (ii), (iii), (iv) with the help of the Chinese remainder theorem yield, respectively, the solutions \( x \equiv 71 \equiv 85 \equiv 6 \equiv 20 \pmod{91} \). Since \( x \equiv 85 \equiv -6 \pmod{91} \) and \( 71 \equiv -20 \pmod{91} \) we have the following solutions \( x \equiv \pm 6 \pmod{91}, x \equiv \pm 20 \pmod{91} \). A has to pick one of these solutions to give to B.

**Step 4:** If A sends 6, then B adds 20 to 6 to get 26 and using the Euclid’s algorithm computes \( \gcd(26, 91) = 13 \). B divides 91 by 13, to 7 (the other factor) and B wins the toss. On the other hand, if B receives 20, he will not be able to factorize 91 and A wins the toss. Therefore, there is an equal chance (just like that of actual coin tossing) for B to receive a solution that helps B rapidly factor 91, or a solution that does not help him factor.

**General algorithm**

In general terms this algorithm goes as follows:

- **Step 1:** A chooses two primes \( p \) and \( q \) and calculates \( n = pq \). A sends \( n \) to B.
- **Step 2:** B chooses a random number \( x \) smaller than \( n \) and calculates the least remainder of \( x^2 \) modulo \( n \). B sends \( <x^2>_n \) to A.
- **Step 3:** With the knowledge that \( p \) and \( q \) are primes and remembering the Chinese remainder theorem, A calculates the four square roots \( \pm x \) and \( \pm y \). A sends one of these four values to B, not knowing which one B had originally chosen in step 2.
- **Step 4:** If A sends \( +x \) or \( -x \), B has lost, admits it, and A wins the toss. However, if A sends \( +y \) or \( -y \), then B wins the toss.

**In the event of an argument:**

Suppose one party does not believe the other; for instance, suppose A does not believe that B has won and seeks for a proof, the protocol continues.

- **Step 5:** B adds his \( x \) to the \( y \) sent by A and with the Euclid’s algorithm, calculates \( \gcd(x + y, n) = p \) or \( q \). B sends \( \gcd(x + y, n) \) and \( n/\gcd(x + y, n) \) to A, that is, the two factors \( p \) and \( q \) of \( n \). A knowing that B has succeeded in factoring \( n \), must have to succumb.

**References**


Received: April 20, 2006