Short Summary of Mathematical Results for Cryptanalysis

John Manferdelli, University of Washington

September 25, 2008

1 Introduction

I've collected a number of results here that I might use implicitly or explicitly. Most of the time, I will go over the results before I use them and maybe even do the proofs but in case I forget, you can always refer to this. This also serves the purpose of establishing (or at least alerting you to) some notation that might be used.

This description is pretty compact and I'd bet a large sum of money that there are missing definitions. Unless you are much smarter than any person I've ever met, you should not expect to be able to understand any portion of this for a topic you've never seen before in your life.

During the year, I will happily cover in class any material that people are unfamiliar with and the refernce section at the end points to expositions that are far more readable than this is, so relax.

2 Mathematics and Cryptanalysis

Shannon succintly explained the relationship between Mathematics and Cryptanalysis in his famous "Mathematical Theory of Secrecy Systems." He said "Breaking a good cipher should require as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type." This is still true and that's why mathematics is so important for cryptanalysis. Mathematics is a good way to understand things generally and is fun so what more do you need? [1]

3 Sets, Mappings and Counting

Let A and B be sets and $f : A \to B$. $Im(f) = \{f(x) : x \in A\}$. f is *surjective* if $Im(f) = B$; in this case, f is called a surjection. f is *injective* if $f(x) = f(y) \to x = y$; in this case, f is called a injection. f is *bijective* if it is both injective and surjective; in this case, f is called a bijection. The empty set is denoted by $\emptyset$. The *cardinality* of a set A is denoted by $|A|$. When A has a finite number of elements, $|A|$ is just the number of elements in A; obviously, $|\emptyset| = 0$. If f is a bijection between A and B, $|A| = |B|$. We say A has n elements if there is a bijection between A and the set $\{1, 2, \ldots, n\}$.

If A is finite with N elements ($|A| = N$), the set of bijections $\sigma : A \to B$ is denoted by S_N; it turns out, this set forms a group under function composition and this group is called the *symmetric group* on N elements. $|S_N| = N! = N(N-1)\ldots 1 = \prod_{i=1}^{N} i$. There are $N!$ bijections of a set onto itself; by contrast, there are N^N functions mapping a set to itself.

As usual, $A \cap B$ is the *intersection* of A and B and $A \cup B$ is the *union* of A and B.

[1] I am told that Simone Weil (the sister of Andre Weil, a famous mathematician) who was a very influential writer in the early twentieth century, is quoted as saying "There are only two important things in the world: sex and mathematics." Since the Weils famously had both covered, this might have been understandable family pride but more likely was intended to suggest that both were fundamental, or fun, or something. Actually, both Weil's are a little hard to understand sometimes.

4 Number Theory

A number $p \in \mathbb{Z}^{>0}$ is *prime* if whenever $a, b \in \mathbb{Z}^{>0}$ and $ab = p$, either $a = p$ or $b = p$. An absolutely fundamental property of the integers is the division property expressed in the *Division Algorithm:* If $a, b \in \mathbb{Z}, a > b, \exists q, r \in \mathbb{Z}^{\geq 0}$ such that $a = bq + r, r < b$. Let $g \in \mathbb{Z}^{>0}$ be the smallest positive number such that $g = ai + bj, i, j \in \mathbb{Z}$ then (1) $g \mid a$, $g \mid b$; and, (2) if $s \in \mathbb{Z}$ with $s \mid a$ and $s \mid b$ then $g \mid s$. g is called the greatest common denominator and this is denoted $g = (a, b)$. We say a and b are co-prime or relatively prime if $(a, b) = 1$. This is proved using the Division Algorithm and, in fact, the Division Algorithm can be repeatedly applied in recipe called *Euclid's Algorithm* to calculate $g, i, j : g = (a, b) = ai + bj$. Euclid's algorithm is surely one of the most important in computer science.

We can use the the results above to the **Fundamental Theorem of Arithmetic** that every positive integer is a product of primes in an essentially unique way. Further, we can use the procedure to solve linear congruences and to prove the Chinese Remainder Theorem.

With respect to solving linear equations mod m, let's start with the case that the modulus is prime. We want to find a solution to $ax = b \pmod{p}$. If $a = 0 \pmod{p}$, there is no solution unless $b = 0 \pmod{p}$ in which case everything is a solution. Assume now that $a \neq 0 \pmod{p}$ then $(a, p) = 1$ (since p is prime) and $\exists u, v \in \mathbb{Z} : au + pv = 1$. Reducing mod p, we get $au = 1 \pmod{p}$. Multiplying both sides by b, we get $a(ub) = b \pmod{p}$ so the solution is ub. Notice that u is the multiplicative inverse of $a \pmod{p}$; in practice, this is how an inverse mod p is computed. If the modulus, m, is *not* prime, there is a solution to $ax = b \pmod{m}$ only if $(a, m) = g \mid b$. In that case, just solve the equation $\frac{a}{g}x = \frac{b}{g} \pmod{\frac{m}{g}}$ to recover a solution to the original equation. More sophisticated techniques are required for solving non-linear equations over prime power moduli. Once you can solve over prime powers, you can use the Chinese Remainder Theorem to get a solution over any modulus.

Suppose we want to solve and equation over a composite modulus $m = m_1 m_2$ where $(m_1, m_2) = 1$. We can do this by solving the equations over m_1 and m_2 seperately (As we saw, this is often easy if the m_i are primes.) and combining the solutions with the *Chinese Remainder Theorem.* **Chinese Remainder Theorem:** If $(m_1, m_2) = 1$ and $x = a_i \pmod{m_i}$, $i = 1, 2$, then these equations have a common solution that is unique mod m. Here is a procedure that carries out the program suggested by the theorem: Find (Euclidean Algorithm!) $u, v \in \mathbb{Z} : um_1 + vm_2 = 1$. Take a look at $y = ua_2m_1 + va_1m_2$. $y = ua_2m_1 \pmod{m_2}$, $um_1 \pmod{m_2} = 1$, $y = va_1m_2 \pmod{m_1}$, and $vm_2 \pmod{m_1} = 1$, so y is exactly the solution we are looking for; it is easy to check that it is unique mod m.

Solving non-linear equations is quite a bit harder than solving linear equations. The simplest non-linear problem is to solve $x^2 = a \pmod{m}$ and even this problem is so hard that it gives rise to cryptographic systems that exploit the "hardness" when m is composite. When m is a prime, there is a nice theory that will be useful in number theory based cryptosystems. I will mention the main results. First a definition: If $(a, m) = 1$, the number a is called a quadratic residue if the equation $x^2 = a \pmod{m}$ has a solution. If this equation has a solution, it has two since $-x$ is another one. Let p be an odd prime and define the "Legendre symbol" as follows: $(\frac{a}{p}) = 1$ if a is a quadratic residue mod p and $(\frac{a}{p}) = -1$ otherwise unless $a = 0 \pmod{p}$ in which case $(\frac{a}{p}) = 0$. Here are two basic rules for computing the Legendre symbol: Assume $(a, p) = 1 = (b, p)$ then $(\frac{a}{p}) = a^{\frac{p-1}{2}} \pmod{p}$ and $(\frac{ab}{p}) = (\frac{a}{p})(\frac{b}{p})$. The central result in the theory of quadratic residues is a beautiful theorem of Gauss called the *law of quadratic reciprocity*: if p, q are odd primes then $(\frac{p}{q})(\frac{q}{p}) = (-1)^{\frac{(p-1)}{2}\frac{(q-1)}{2}}$. The Legendre symbol can be extended to non-prime moduli where it's called the Jacobi symbol.

The *Prime Number Theorem* is important for cryptography since it describes the "density" of primes in the integers. We will *not* prove it, since even the shortest, simplest proof is long and involves techniques from complex analysis. **Prime Number Theorem:** Let $\pi(x)$ denote the number of primes $\leq x$ then $lim_{x \to \infty} \frac{\pi(x) ln(x)}{x} = 1$. In fact, the accuracy of the asymptotic approximation is pretty good, pretty fast.

5 Groups, Permutation Groups

A set G is a group under the binary operation "$\cdot$" if it has the following properties: (1) $\exists e \in G : \forall x \in G, e \cdot x = x \cdot e = x$ (existence of identity); (2) $\forall x, y, z \in G : (x \cdot y) \cdot z = x \cdot (y \cdot z)$ (associativity); and, (3) $\forall x \in G, \exists x^{-1} \in G : x \cdot x^{-1} = x^{-1} \cdot x = e$ (existence of inverse). Note that it is *not* required that $x \cdot y = y \cdot x$ although some groups have this property; those groups are called *commutative*. A group is finite if it has a finite number of elements. Often we omit the $\cdot$ understanding that xy means $x \cdot y$.

You're familiar with many groups already. For example, the integers, $\mathbb{Z}$ form an infinite commutative group under the operation $+$ with $e = 0$. The integers mod n, $\mathbb{Z}_n$, form a finite commutative group with n elements under the operation $+$ with $e = 0$. If we denote by $\mathbb{Z}_n^*$ the set of elements in $\mathbb{Z}_n$ which have multiplicative inverses (i.e. $\{x \in \mathbb{Z}_n : \exists y \in \mathbb{Z}_n : xy = 1 \pmod{n}\}$) then $\mathbb{Z}_n^*$ is a group *under the binary operation of multiplication* mod n with identity $e = 1$. By the way, the size of this group is $\phi(n)$ which is the number of integers less than n which are co-prime to n. If $n = \prod_{i=1}^{k} {p_i}^{e_i}$, $\phi(n) = n \prod_{i=1}^{k} (1 - \frac{1}{p_i})$. This is actually an important result in *number theory* that we could have covered, from a different perspective, elsewhere.

H is a *subgroup* of G, written $G \geq H$, if $H \subseteq G$ and H is a group under the same binary operation as G. As you know, $x^m, m \in \mathbb{Z}^{\geq 0}$ is defined recursively as follows: $x^0 = e$ and $x^{m+1} = x \cdot x^m$; $x^{-m} = (x^m)^{-1}$. If n is the smallest non-negative integer such that $x^n = e$, we say x has order n or $|x| = n$.

Although infinite groups are important, in cryptography we often do calculations in finite groups. In finite groups, every element has finite order. Proof: if $x = e$, $|x| = 1$; let $e \neq x \in G$. Since G is finite, the sequence $x, x^2, \ldots, x^n, \ldots$ has repeats. Let $i > j$ be two integers with shortest positive difference such that $x^i = x^j$. Multiplying both sides of this equation by x^{-j}, we get $x^{i-j} = e$, $|x| = i - j$.

The first important theorem in group theory is the *Theorem of Lagrange:* If G is finite and $G \geq H$, then G is the disjoint union of sets of the form Hg and thus $|H| \mid |G|$. Lagrange's theorem is proved by showing that the sets $Hg = \{hg : h \in H\}$ are either disjoint or identical; these sets are called cosets. The theorem easily follows. We can use Lagrange's theorem to show: $|x| \mid |G|$. Proof: let $H = \{x^i\}, i = 1, 2, \ldots\}$. You can check that H is a subgroup called the cyclic group generated by x (sometimes we say $H =< x >$) and $|H| = |x|$. Applying Lagrange, we get $|x| \mid |G|$.

Because cryptography concerns, in part, "reversible" transformations between message blocks of the same size, the bijections of a finite set of N elements is very important. These bijections give rise to a very important group is S_N which consists of all bijections from a set of N elements to itself. In cryptography, bijections usually act on n bit messages. In this case, the "message space" has $N = 2^n$ elements. Let σ be an element of S_N and the set being "permuted" be $X_N = \{1, 2, \ldots, N\}$. σ can be completely specified by its effect on each element of X_N, $1 \mapsto a_1$, $2 \mapsto a_2$, ..., $N \mapsto a_N$. This can be represented pictorially as $\sigma = \begin{pmatrix} 1 & 2 & \ldots & N \\ a_1 & a_2 & \ldots & a_N \end{pmatrix}$. For $N = 5$, for example, we might have $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 1 & 5 & 4 \end{pmatrix}$. Another way to describe σ is in "cycle structure" form where we successively apply σ until the result "wraps around." For the simple example above, $\sigma = (123)(45)$ in cycle structure form. The cycle structure can be obtained from by picking $x \in \{1, 2, \ldots, N\}$ and forming $(x, \sigma(x), \sigma^2(x), \ldots \sigma^i(x))$ where $\sigma^{i+1}(x) = x$ and then pick a y that is not in this cycle and forming it's cycle, etc, until all N elements appear. In this representation, it is common not to list the singleton cycles (x) where $\sigma(x) = x$. The nice thing about cycle structure is that the element order is easy to figure out: if the cycle representation is $\sigma = (a_{1,1}, a_{1,2}, \ldots, a_{1,i_1})(a_{2,1}, a_{2,2}, \ldots, a_{2,i_2}) \ldots (a_{k,1}, a_{k,2}, \ldots, a_{k,i_k})$, then $|\sigma| = LCM[i_1, i_2, \ldots, i_k]$. For example, $|(123)(45)| = 6$.

To "multiply" two permutations, we need to compose the maps. There are two conventions for describing multiplication for composition of maps, "from the left" and "from the right." In "from the right" style, we compute $\rho = \sigma_1 \cdot \sigma_2$ by mapping an element x through σ_2 first and then applying σ_1 to $\sigma_2(x)$; so $\rho(x) = \sigma_1(\sigma_2(x))$. This of course completely specifies ρ. The same operation done "from the left" would be denoted $((x)\sigma_2)\sigma_1 = (x)\rho$ which may seem a little less familiar. It's important to pick one convention and stick with it in calculation because, since S_N is *not* commutative for $N > 2$, $\sigma_1 \cdot \sigma_2 \neq \sigma_2 \cdot \sigma_1$. You'll see both conventions used in cryptography papers.

Let G, H be two groups and $\varphi : G \to H$ be a map. For simplicity, we denote the group operation as "$\cdot$" in both G and H even though they are *not* generally the same operation. φ is a homomorphism if $\varphi(a \cdot b) = \varphi(a) \cdot \varphi(b), \forall a, b \in G$. A homomorphism which is surjective as a map is a surjective homomorphism or epimorphism and one that is injective as a map is an injective homomorphism or monomorphism. A homomorphism which is injective and surjective as a map is called an isomorphism. Two groups G and H are "isomorphic" if the elements of one correspond to a elements of the other in a way that preserves the action of the binary operation, that is, $\varphi(ab) = \varphi(a)\varphi(b)$. If two finite groups are isomorphic, denoted $G \cong H$, they have the same number of elements and if $\varphi(a) = b$, a and b have the same order.

If φ is a homomorphism from $G \to H$, we can define a set (which turns out to be a subgroup) of G called the kernel by $ker(\varphi) = \{x : \varphi(x) = e\}$. In fact, $ker(\varphi)$ is a special kind of subgroup called normal. N is a *normal* subgroup of G if it is a subgroup and it has the additional property that $y^{-1}Ny = N, \forall y \in G$, we denote this by $G \triangleright N$. Normal subgroups are important because they allow us to decompose subgroups in a useful way. Suppose $G \triangleright N$, we define the "factor group" as the set or cosets $\{Na\}$ together with the inherited operation $(Na) \cdot (Nb) = N(a \cdot b)$. We denote the factor group by G/N; note that $|G/N| = |G|/|N|$. The three **isomorphism theorems** are (1) If $\varphi : G \to H$ is a homomorphism, $G/ker(\varphi) \cong Im(\varphi)$, (2) If $G \triangleright H$ and $G \triangleright N$ and $N \subseteq H \subseteq G$ then $G/H \cong (G/N)/(H/N)$, (3) If $G = HN$, $G \triangleright N$ then $HN/N \cong H/(H \cap N)$.

S_N has a normal subgroup A_N which consists of all the "even" permutations S_N that is permutations, σ, for which $\prod_{i<j} \frac{x_{\sigma(i)} - x_{\sigma(j)}}{x_i - x_j} = 1$. $|S_N/A_N| = 2$.

We're going through things fast but I can't resist mentioning one of the major results of twentieth century mathematics. If a group has no non-trivial normal subgroups, it is called *simple*. The cyclic group $< x >, |x| = p$ with p a prime is simple as is alternating group $A_n, n > 4$. If a group is not simple, it contains a normal subgroup and knowing the structure of N and G/N dictates many of the properties of G so if we understood the "atoms" of finite groups, the simple groups, we'd know an awful lot about *all* possible finite groups. The great theorem of finite group theory was the classification of all finite simple groups into a few dozen infinite families along with 26 (God knows what is special about 26.) sporadic simple groups which fall into no systematic family.

If G is a group and $x_1, x_2, \ldots, x_k \in G$ define $H =< x_1, x_2, \ldots, x_k >$ as the smallest subgroup in G containing $x_1, x_2, \ldots, x_k$. Equivalently H consists of all possible products of $x_1, x_2, \ldots, x_k$ and their inverses. A theorem which provides intuition about the kinds of subgroups in S_N, generated by two elements chosen at random, is **Netto's Conjecture** proved by Dixon in the following form: The proportion of elements of $x, y \in S_N$ that generate S_N tends to $\frac{3}{4}$ as $n \to \infty$.

6 Algebra

The basic results of algebra and a familiarity with rings and modules (and their morphisms) are useful but I will not cover them exhaustively. In fact, I'll mainly focus on finite fields, polynomials (mostly solving them) and linear algebra. Linear algebra is important because, in the end, we can only efficiently solve linear equations and even the "tricks" that accelerate solving non-linear equations often end up using linear algebra. We'll talk about that in the section after the present one.

We're all familiar with the mathematical objects called fields. A field, F, is a commutative group with respect to an additive operation, "+" and $F - \{0\}$ is commutative group with respect to the multiplication operation "$\cdot$". Fields are required to have a further property relating the two operations, namely, the distributive law $a \cdot (b + c) = a \cdot b + a \cdot c$. The most familiar fields are the rational numbers R, the real numbers, $\mathbb{R}$ and the complex numbers $\mathbb{C}$. The complex numbers are a little special since they are algebraically complete; that is, every polynomial equation, $p(x) = 0$ in $\mathbb{C}$ has a solution (called a *root* of $p(x)$) in $\mathbb{C}$ and hence by the division algorithm we discuss later, all solutions lie in $\mathbb{C}$. This result is called *Fundamental Theorem of Algebra*. The real numbers and complex numbers are also *complete*; that is, every Cauchy sequence converges to an element in the underlying field.

Two related, less restricted, algebraic concepts are commutative rings and modules. A *commutative ring* has the basic properties of a field *except* non-zero elements are *not* required to have inverses. A *module*,

M, over a ring R, is a commutative group under the $+$ operation and posseses a composition operation $(R, M) \to M$ (again we call this "$\cdot$" even though it is different from the $\cdot$ operation in R) with the following properties: $r \cdot (m_1 + m_2) = r \cdot m_1 + r \cdot m_2$, $(r_1 + r_2) \cdot m = r_1 \cdot m + r_2 \cdot m$, and $1 \cdot m = m$. We sometimes call M an $R-$module to emphasize the role of R. A vector space V of finite dimension is a module. real numbers (viewed as a ring) acting on a vector So the usual three dimensional space, V, is simply an $\mathbb{R}$-module.

Rings are modules over themselves. The reason rings (and modules) are so important is they can be decomposed in iteresting ways that illuminate the structure of the parent object. Here is such construction that we will use to construct finite fields. Let R be a ring and $x \in R$, the set of all R-multiples of x denoted by $I = Rx$ is an module over R. I is also an *ideal* of R (a special kind, in fact). An ideal of a ring R is a subgroup of the additive group of R (i.e.- is closed under addition) and satisfies the property $x \in I \to rx \in I$. If we define the factor object R/I whose elements are the cosets of I in R regarded as a commutative group under addition and define the operations $a+I+b+I = (a+b)+I$ and $r(a+I) = ra+I$, this factor object is also a ring. Here's an example that ties this together and shows the importance of factor objects. Consider $\mathbb{Z}$ as a ring and $I = \{np : n \in \mathbb{Z}\}$ with p prime. Convince yourself that I is an ideal of $\mathbb{Z}$. Look carefully and you'll notice that $\mathbb{Z}/I$ is isomorphic to the ring $\mathbb{Z}_p$ or the integers modulo p. We can also define factor module in the obvious way. Ideals generated by a single element, like Rx, are called principal ideals and are often denoted by (x) when R is clear.

R, the rational numbers, $\mathbb{R}$ and $\mathbb{C}$ are all fields of characteristic 0 meaning that $n \cdot 1 \neq 0$ for any positive integer n. Fields which do not have this property have characteristic p for some prime, p (Why must a finite characteristic be prime?). A familiar example of a field of characteristic p is the integers modulo p. This is a finite field with p elements where $p \cdot 1 = 0$. There are infinite fields of characteristic p and, in fact, any algebraically complete field of characteristic p will be infinite. However, we will mostly be concerned with finite fields of characteristic p called *Galois fields*.

Galois fields can be constructed by starting with a finite field of characteristic p and "extending it" so that it contains a root of a polynomial that was irreducible in the original finite field. Extending a field is an interesting and important operation theoretically and computationally. Here is an example of such a construction which is easy to generalize: Let $GF(2)$ be the field of integers mod 2. This is the smallest possible field and has characteristic 2 ($1+1 = 0$ in $GF(2)$). $GF(2)$ has only two elements: 0 and 1. Consider the *irreducible* polynomial $g(x) = x^2 + x + 1$ in the ring $GF(2)[x]$[2] and its associated ideal $(g(x))$ consisting of all $GF(2)[x]$-multiples of $g(x)$. The factor object $\frac{GF(2)[x]}{(g(x))}$ has four elements. It is a field with four elements of characteristic 2. More generally, if $g(x)$ is an irreducible polynomial of degree d, the extension field would have 2^d elements. All finite fields can be constructed this way. The fact that g was irreducible is required to show the non zero elements in the factor object [which can be represented as elements of the form $y = a + bx, a, b, \in GF(2)$] have inverses. We'll do that when we talk about polynomials below.

If F is a finite field, the multiplicative group of F is cyclic and any generator of $F - \{0\}$ is called a primitive element. I won't prove this. One can also count the primitive roots, for example, $\mathbb{Z}_p$ had $\phi(p-1)$ primitive roots. I won't prove that either. Another important fact is that if F is a finite field *every* function from $F \to F$ can be represented as a polynomial over F. To show this, suppose f is a function with values $f(a_i) = b_i$ and put $P_a(x) = \prod_{\alpha \in F, \alpha \neq a} \frac{(x-\alpha)}{(a-\alpha)}$. There are finitely many such a_i's and we can set $f(x) = \sum_{a \in F} f(a) P_a(x)$. It is easy to show this polynomial has precisely the right values. As a concrete example, consider $GF(3)$ the integers mod 3 regarded as a finite field with 3 elements $0, 1, 2$ and let f be specified (in full generality by $f(0) = a_0, f(1) = a_1, f(2) = a_2$. $g(x) = a_0 \frac{(x-1)(x-2)}{(0-1)(0-2)} + a_1 \frac{(x-0)(x-2)}{(1-0)(1-2)} + a_2 \frac{(x-1)(x-0)}{(2-0)(2-1)}$.

Now let's investigate polynomials, particularly polynomials over finite fields. Polynomials are elements of $F[x]$; they are things of the form $p(x)$ where $p(x) = \sum_{i=0}^{n} a_i x^i$. Just like integers, polynomials obey a division algorithm: if $a(x)$ and $b(x)$ are two polynomials with $deg(a(x)) \geq deg(b(x))$ then $\exists q(x), r(x)$ such that $a(x) = b(x)q(x) + r(x)$ where $deg(r(x)) < deg(b(x))$. As a result, although we haven't proved this, just like the integers, every ideal in $F[x]$ is principal and, just like the integers, the ring of polynomials in a single variable, x, is a *unique factorization domain* or UFD. A UFD is a ring with cancellation (no zero divisors) in which every element can be written as the product of irreducible elements (primes) in an essentially unique

[2] That may have gone by too fast. $F[x]$ is the ring consisting of all polynomials in x with coefficients in F. Convince yourself this is a ring. In the present example, $F = GF(2)$.

way. Restating this, $F[x]$ is a "UFD". Wait, there's more: if U is any UFD, $U[y]$ is a UFD. So not only univariate polynomials are UFDs, multi-variate polynomials (elements of $F[X_1, X_2, \ldots, X_n]$) are UFD's. If $F = \mathbb{C}$, since F is algebraically closed, irreducible polynomials must be linear so any polynomial, $g(x)$, in $\mathbb{C}[x]$ can be factored as $g(x) = a(x - a_1)(x - a_2)\ldots(x - a_n)$.

Often, we want to factor polynomials into irreducible elements. Here is a simple observation you learned in high school that uses the division algorithm for polynomials: If a is a root of $p(x)$, $(x-a) \mid p(x)$. Another way of saying this is $p(x) = (x - a)q(x)$ where $q(x)$ is another polynomial. The proof is simple. By the division algorithm for polynomials, $p(x) = (x-a)q(x)+r(x)$ where $deg(r(x)) < deg(x-a) = 1$. The inequality forces $r(x)$ to be a constant, say b and $p(x) = (x - a)q(x) + b$. Substituting $x = a$, we get $p(a) = (a - a)q(a) + b$. However, a is a root so $p(a) = 0 = 0(q(a)) + b = b$ and thus $p(x) = (x - a)q(x)$. Continuing in the same vein, we can show that $p(x)$ has no more that $deg(p(x))$ roots and, in fact, $p(x)$ has exactly $deg(p(x))$ roots (counting multiplicity) if the underlying field is algebraically closed.

Now we can fulfill our promise to show that non-zero elements of $F[x]/(g(x))$ have inverses if $g(x)$ is irreducible: Suppose $f(x) \neq 0$, then the gcd, $h(x)$, of $f(x)$ and $g(x)$ can be expressed as $h(x) = a(x)f(x) + b(x)g(x)$. Since $h(x) \mid g(x)$ and $g(x)$ is irreducible, either $h(x)$ (and hence $f(x)$) must be a multiple of $g(x)$ or $h(x)$ has degree 1 [3] and hence is a non-zero constant c. In the former case, $f(x) \in (g(x))$ so $f(x) = 0$ as a factor object and there is nothing to prove. In the later case, dividing by c, we get $1 = f(x)(\frac{1}{c}a(x))+g(x)(\frac{1}{c}b(x))$. Reducing mod $g(x)$, we get $1 = f(x)(\frac{1}{c}a(x)) \pmod{g(x)}$ and $(\frac{1}{c}a(x)) \pmod{g(x)}$ is the desired inverse.

The Fundamental Theorem of Algebra describes the relationship between the roots of a polynomial and its degree. Bezout's theorem characterizes the common solutions of two polynomials by their degrees. **Bezout's Theorem:** If two polynomials $f(x), g(x)$ have degrees m, n respectively and have no common component, then $f(x) = 0$ and $g(x) = 0$ intesect in mn points (counting multiplicity).

There are two way to specify a polynomial of degree n. We can specify the $n + 1$ coefficients $a_0, a_1, \ldots, a_n$ or we can specify the values $y_0, y_1, \ldots, y_n$ of $f(x)$ at $n + 1$ points $x_0, x_1, \ldots, x_n$. These representations are related by a matrix called the Vandermonde matrix. In fact,

$$\begin{pmatrix} y_0 \\ y_1 \\ \ldots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_0 & \ldots & {x_0}^{n-1} & {x_0}^n \\ 1 & x_1 & \ldots & {x_1}^{n-1} & {x_1}^n \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_n & \ldots & {x_n}^{n-1} & {x_n}^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \ldots \\ a_n \end{pmatrix}.$$

If the x_i's are distinct, the Vandermonde matrix is invertible so we can solve for the a_i's in terms of the y_i's and vice versa.

There are several ways to solve a set of polynomial equations and obtain common solutions. One technique uses the concept of a resultant. If $f_v(x) = v_n x^n + \ldots + v_0$ and $g_w(x) = w_m x^m + \ldots + w_0$, $\exists \phi_{v,w}(x), \psi_{v,w}(x), R(u, v) : \phi_{v,w}(x)f_v(x) + \psi_{v,w}(x)g_u(x) = R(v, w) = v_m^n w_n^m \prod_{i<j}(t_i - u_j)$, where t_i, u_j are roots of f, g respectively. $R(u, v)$ is the *resultant* and it is 0 iff $f_u(x) = 0$ and $g_v(x) = 0$ have a common solution. To show this, consider the equations written in matrix notation:

$$\begin{pmatrix} x^{m-1}f_v(x) \\ x^{m-2}f_v(x) \\ \ldots \\ f_v(x) \\ x^{n-1}g_w(x) \\ x^{n-2}g_w(x) \\ \ldots \\ g_w(x) \end{pmatrix} = \begin{pmatrix} v_n & v_{n-1} & \ldots & v_0 & 0 & 0 & \ldots & 0 \\ 0 & v_n & v_{n-1} & \ldots & v_0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 & v_n & v_{n-1} & \ldots & v_0 \\ w_m & w_{m-1} & \ldots & w_0 & 0 & 0 & \ldots & 0 \\ 0 & w_m & w_{m-1} & \ldots & w_0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 & w_m & w_{m-1} & \ldots & w_0 \end{pmatrix} \begin{pmatrix} x^{n+m-1} \\ x^{n+m-2} \\ \ldots \\ \ldots \\ \ldots \\ \ldots \\ x \\ 1 \end{pmatrix}$$

Proof: Let the column vectors be $C_{m+n-1} \ldots C_0$. $C = (x^{m-1}f_v(x), \ldots, g_w(x))^T$. $C = C_{m+n-1} \cdot x_{m+n-1} + \ldots + 1 \cdot C_0$. Now solve for 1. $1 = \frac{det(C_{m+n-1} \ldots C_1, C)}{det(C_{m+n-1} \ldots C_1, C_0)}$. Get $\phi_{v,w}(x)f_v(x) + \psi_{v,w}(x)g_w(x) = R(v, w)$. In principle, we can solve multivatiate polynomials by using resultants to successively eliminate variables.

[3] If it had higher degree, $h(x)$ would be a non-trivial factor of the irreducible $g(x)$ which can't happen.

Another technique for solving simultaneous polynomial equations involves "Grobner Basis" but we won't cover that here.

7 Linear Algebra

I realize we have not really followed a strictly linear order of presentation and, in fact, we've used results from linear algebra in prior sections. Sorry, General Bourbaki.

Linear algebra studies the transformations $L : V \to V'$ between vector spaces (and more generally modules) with the property that $L(x + y) = L(x) + L(y)$ where $x, y \in V$. These maps are morphisms from V to V'. It is very important to observe that one can and we will study linear algebra over finite fields not just over real or complex numbers.

If V is a vector space over F, with basis $< v_1, v_2, \ldots, v_n >$ and L is a linear transformation on V, L can be specified by its effect on each of the basis vectors. If $L(v_i) = a_{i1}v_1 + a_{i,2}v_2 + \ldots + a_{in}v_n$, for $i = 1, 2, \ldots, n$, the coefficients a_{ij} specify the linear transformation, given the basis, and can be organized as a matrix. That's how matricies are born.

One theme related to algebraic structures we've mentioned was that of decomposition into sub-structures which can be simply reassembled to produce the properties of the original sturcture. Linear algebra is no exception, and the crucial property required to do this is invariance. From now on we assume $L : V \to V$. A subspace, W, is invariant under L iff $L(W) \subseteq W$. Nothing we've said so far requires L to be invertible. However, if $L : V \to V$ is a surjection, it is an injection and if it is an injection, it is a surjection. If L does *not* induce a vector space isomorphism, there are two non-trivial invariant subspaces that can be obtained from L. The first is the image of V under L and the second, is the null space or kernel of L defined by $ker(L) = \{v \in V : L(v) = 0\}$. In fact (and I won't prove it here), $V \cong Im(L) \oplus ker(V)$ which leads to the result that the dimension of V as a vector space is the sum of the dimensions of $ker(L)$ and $Im(L)$ as vector spaces. If $ker(L) = 0$, L is a bijection and is invertible. The inverse of an invertible linear function is itself linear and the set of linear bijections is a subgroup of all bijections on V. In the case that V is finite (which happens when F is finite and V is finite-dimensional over F), the group of linear transformations is is very interesting in its own right and we will calculate its size below. When we decompose a vector space into the direct sum of invariant subspaces under L, this induces a natural change in the basis and matrix representation of the linear transformation. We study that next.

Let $[e] = \{e_1, \ldots, e_n\}$ be a basis for V and let L be a linear transformation on V. Let $v_{[e]} = [c_1, c_2, ..., c_n]^T$ denote the coordinates of v with respect to $[e]$: $v_{[e]} = c_1e_1 + ... + c_ne_n$. Let $L_{[e]}$ denote the matrix for L with respect to $[e]$: $L_{[e]} = \sum_j a_{ji}e_j$. Then $L_{[e]}v_{[e]} = (Lv)_{[e]}$. If $f_i = \sum_j b_{ji}e_j$ is another basis, $P = (b_{ij})$ is called the transition matrix from $[f]$ to $[e]$ and P^{-1} is the transition matrix from $[e]$ to $[f]$ (note the sum over the first index). $Pv_{[f]} = v_{[e]}$ and $v_{[f]} = P^{-1}v_{[e]}$. Finally, $L_{[f]} = P^{-1}L_{[e]}P$. The same holds over free modules (I know, I haven't defined free modules). This shows that the conjugates of the matrix, M, matricies of the form $N^{-1}MN$, represent the self same linear transformation expressed in terms of another basis and two matricies related in that way are said to be *similar*. Note that representing a linear transformation on a direct sum of two invariant subspaces simplifies the appearance of the matrix in this new basis. In the new basis, the matrix is divided into four rectangular blocks with non-zero entries in only the upper left and lower right blocks.

There are two prototypical invariant subspaces which stand out as "natural" when regarding linear transformations. Suppose $L(v) = \lambda v, v \neq 0$. The subspace $L(v)$ is obviously L-invariant. In fact, *if we can find n linear independant such vectors, where* $n = dim(V)$, the matrix for L in the basis consisting of these vectors is diagonal with entries $\lambda_1, \lambda_2, \ldots, \lambda_n$ along the diagonal and 0 elsewhere. We denote this matrix as $diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$. This is called a spectral decomposition and is it is the best possible decomposition in terms of simplifying the matrix and making the properties of the linear transformation obvious. If such a decomposition exists, there exist linear transformations $E_1, E_2, \ldots, E_n$ such that $E_i^2 = E_i$ and $E_iE_j = 0$.

Unfortunately, not all transformations are diagonalizable. Matricies can fail to be diagonalizable because the eigenvalues, λ_i, fail to lie in the field F (this can't happen if the field is algebraically closed). However,

there are matricies which are just not diagonalizeable for more fundamental reasons. Fortunately, we can decompose all matricies into subspaces in which the matrix is simple enough to get a lot of information even if it is "not quite as simple" as a spectral decomposition. Each of these decomposition is called a "canonical form" of the transformation because if we have two matricies, they are similar if and only if they have the same canonical form.

Another "obviously" L-invariant subspace of V can be constructed as follows: Let $v \neq 0$ and consider the sequence of spaces $< v >, < v, L(V) >, \ldots, < v, L(v), L^2(v) \ldots L^{k-1}(v) >$ where each space is larger than the last. This construction continues until $L^k(v)$ is in the space generated by the previous images of powers of L, when this happens, $L^k(v) = (a_{k-1}L^{k-1} + a_{k-2}L^{k-2} + \ldots + a_0)v$. Let $W =< v, L(v), L^2(v) \ldots L^{k-1}(v) >$ be the subspace of dimension k resulting from the foregoing construction. A *minimal* polynomial of a linear transformation in a vector space V is one that results in 0 when the polynomial, with the transformation substituted is applied to any vector in V and is denoted by $m_{L(V)}(x)$. So $m_{L(W)}(x) = x^k - (a_{k-1}x^{k-1} + a_{k-2}^{k-2} + \ldots + a_0)$. It is clear that $L(W) \subseteq W$ and in fact $L^k - (a_{k-1}L^{k-1} + a_{k-2}L^{k-2} + \ldots + a_0)w = 0, \forall w \in W$. If $W = W_1 \oplus W_2$ is a decomposition into two L-invariant subspaces $m_{L(W)}(x) = [m_{L(W_1)}(x), m_{L(W_2)}(x)]$. For any linear transformation, L, on V, we can always find a basis for V such that the matrix representation for L is upper triangular [4] and using this, it is easy to show that the minimal polynomial of any linear transformation on a vector space V of dimension n has degree less than or equal to n.

Now back to diagonalization. It turns out, a linear transformation is diagonalizable if and only if its minimal polynomial is the product of linear factors: $m_L(x) = (x - \lambda_1)(x - \lambda_2) \ldots (x - \lambda_t)$ [5] and two linear transformations are diagonalizeable if both their minimal polynomials are products of linear factors and they commute. I'll assume you know what a determinant is. The characteristic polynomial of a matrix, A, denoted by $c_A(x)$ is $det(A - xI)$ and $c_A(x) = 0$ is called the characteristic equation of the matrix A. Since all matricies representing the same linear transformation are conjugate and since $det(B^{-1}AB) = det(A)$, we can actually calculate $c_A(x)$ but refer to it and the characteristic polynomial of the linear transformation L, $c_{L(V)}$, provided L happens to have matrix A representation in some basis. In general, the minimal polynomial divides the characteristic polynomial but they don't have to be the same. Here is the prototypical example: Set $A = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{pmatrix}$ and $B = \begin{pmatrix} \lambda & 0 & 0 \\ 1 & \lambda & 0 \\ 0 & 1 & \lambda \end{pmatrix}^T$. A and B have the same characteristic polynomial, $(x - \lambda)^3$, but A has minimal polynomial, $(x - \lambda)$, while B has minimal polynomial, $(x - \lambda)^3$.

Okay, now we can fully characterize linear transformations $L : V \to V$ by one of two canonical forms. First, supose that the underlying field is algebraically closed, the minimal polynomial of a linear transformation, L, can be factored as $m_{L(V)}(x) = \prod_{i=1}^{k}(x - \lambda_i)^{n_i}$. If all the $n_i = 1$, the transformation is diagonalizable and can be decomposed into invariant subspaces whose minimal polynomial on each subspace is of the form $(x - \lambda_j)$. Even when all $n_1 \neq 1$, the vector space can be decomposed into invariant subspaces with minimal polynomials $(x - \lambda_i)^{n_i}$. If $n_i > 1$, say $n_i = 3$, the matrix on the corresponding subspace looks like $\begin{pmatrix} \lambda & 0 & 0 \\ 1 & \lambda & 0 \\ 0 & 1 & \lambda \end{pmatrix}$. This decomposition is called *Jordan Cannonical Form* and two matricies are similar if and only they have the same Jordan Canonical Form. If the underlying field, F, is not algebraically closed, the vector space can be decomposed into invariant subspaces whose minimal polynomials are powers of irreducible polynomials (but ones not quite as simple as $(x - \lambda)$). On each of these invariant subspaces, where the irreducible polynomial is of the form $m(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \ldots + a_1x + a_0$, the submatrix will have the form

$$\begin{pmatrix} 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \\ 0 & 0 & 0 & \ldots & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \ldots & -a_{k-2} & -a_{k-1} \end{pmatrix}.$$

When matricies are decomposed into invariant subspaces in this manner the resulting matrix after transforming to the new basis is said to be in *Rational Canonical Form*. Again, two matricies are similar if and

[4] This is the Cayley-Hamilton Theorem.

[5] The λ_i are the eigenvalues we mentioned before.

only they have the same Rational Canonical Form. Notice the spectral decomposition is the same (when it exists) as the corresponding canonical forms.

You probably know that the transpose of a square matrix, A^T, is the matrix obtained by making the rows of A, the columns of A^T. We define a *normal* matrix is one for which $N^T N = NN^T$. If a normal transformation has eigenvalues that are all real, the matrix is hermitian (and hence the matrix is equal to its conjugate transpose) and unitary if $|\lambda| = 1$ for all eigenvalues. The **Spectral Theorem** say A normal operator over an algebraically complete field is diagonalizable. The diagonal elements are called the spectrum and this is called the spectral decomposition.

As promised earlier, let's count the number of invertible transformation, L, from a vector space, V, of dimension n over a finite field F where the number of elements in F is q. If $v_1, v_2, \ldots, v_m$ is a basis for V, v_1 can go into any one of $q^n - 1$ vectors, v_2 can go into any vector which is not a multiple of the first vector, there are $q^n - q$ of these, v_3 can go into $q^n - q^2$ possible vectors, etc.. Thus the total number of transformations is $(q^n - 1)(q^n - q) \ldots (q^n - q^{n-1})$.

8 Probabililty

Let X be a random variable and $Pr(X = x)$ be probability distribution on X. Usually we are interested in probability distribution over a finite number of elements; in that case, $Pr(X = x_i) = p_i$ and $\sum_{i=1}^{n} p_i = 1$. Infinite distributions are also important. There replace "sums" with integrals in what follows.

Given a probability distribution, there are several important parameters that characterize it. The first is the expectation, $E(X)$ where $E(X) = \sum_{i=1}^{n} x_i p_i$. $\mu = E(X)$ is also called the mean. The two other parameters that are important are the standard deviation, $\sigma(X)$, and the variance, $Var(X)$. Both measure the "spread" of likelihood from the mean and are defined by $\sigma^2(X) = Var[X] = E[(X - E[X])^2]$. The *covariance* between two distributions, X, Y is $\mu_{XY} = E((X - X_j)(Y - Y_i))$ and it measures the relationship between two distributions.

A useful approximation, called Stirling's formula, is $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$.

The notation $P(X = a|Y = b)$ means the probability that X takes on the value *a given* that Y, a possibly related distribution finction, takes on the value b (there is a slight abuse of language here which I will cheerfully ignore). $P(X = x, Y = y) = P(X = x|Y = y) = P(Y = y|X = x)$ is the joint distribution of X and Y. X and Y are *independent* if $P(Y = y|X = x) = P(Y = y)$; in which case, $P(Y = y, X = x) = P(X = x)P(Y = y)$. Actually its better to think of A as a set of events and B a set of events whose occurance is governed by some distributions and define $P(A|B)$ as the probability that an event in A occured given that an event in B occured. With this notation, we can state *Bayes Theorem:* $P(B_i|A) = \frac{P(A|B_i)P(B_i)}{\sum P(A|B_j)P(B_j)}$.

There are two very important distributions. The first is the *Normal Distribution* $N(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. This is a continuous distribution over the real numbers with mean μ and standard deviation σ. The second is *Binomial Distribution* $B(N, n, p) = \binom{N}{n} p^n (1-p)^{N-n}$ which describes repeating an experiment of selecting one of two outcomes (the first occurs with probability p, the second,occurs with probability $q = 1 - p$) N times. $B(N, n, p)$ is the probability that the repeated trials resulted in n instances of outcome 1 and $N - n$ instances of outcome 2. Unsurprisingly, we expect to get about Np instances of outcome 1; thus, $E(B) = Np$ and, $\sigma^2(B) = Np(1 - p)$.

The *Central Limit Theorem* justifies our intuition that repeated trials will "tend" towards the correct probability distribution. If X_i are independent, identically distributed random variables with mean μ and standard deviation σ and $S_n = X_1 + \ldots + X_n$, then $\lim_{n\to\infty} P(a \leq \frac{(S_n - n\mu)}{\sigma\sqrt{n}} \leq b) = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-(u^2/2)}$.

Another important relation is *Chebyshev's inequality:* Let Y be a random variable with expected value $\mu_Y = E[Y]$ and variance, $Var(Y)$. Then for any $t > 0$, $\Pr[|Y - \mu_Y| \geq t] \leq \frac{Var(Y)}{t^2}$.

The index of coincidence or "IC" measures the roughness of a distribution. $IC = \sum_{i=1}^{n} \frac{f_i f_{i-1}}{n(n-1)}$.

9 Algorithms

$f \in O(g) \leftrightarrow g \in \Omega(f) \leftrightarrow L_{x\to\infty}\frac{f(x)}{g(x)} < \infty$. $O(f)$ is an upper bound and $\Omega(f)$ is a lower bound. If $g \in O(f) \cap \Omega(f)$ we say $g \in \Theta(f)$ and thus that g is "tightly bounded" by f. $f \in o(g) \leftrightarrow g \in \omega(f) \leftrightarrow L_{x\to\infty}\frac{f(x)}{g(x)} = 0$.

Typically, we wish to analyze the performance of algorithms in terms of the number of "sequential steps" required for exection as a function of the instance input size n. While recursive algorithms naturally lend themselves to analysis by recursion, generally, we would like to obtain a "closed form expression" for $T(n)$. Here are some standard results. Suppose $T(n) = aT(n/b)+f(n)$. If $f(n) = O(n^{log_b(a)-\epsilon})$ then $T(n) = \Theta(n^{log_b(a)})$. If $f(n) = \Theta(n^{log_b(a)})$ then $T(n) = \Theta(n^{log_b(a)}lg(n))$. If $f(n) = \Omega(n^{log_b(a)+\epsilon})$ and $af(n/b) \leq cf(n), c < 1$ then $T(n) = \Theta(f(n))$.

We need to know a little about representing numbers on a computer. Although most "theoretical bounds" on performance of a computation are expressed in terms of bits (i.e.- the input of the computation is expressed, without loss of generality, as a sequence of bits), on most digital computers, data is organized into "bytes" (representing characters) and words, which are a sequence of *wordsize* bits upon which register operations are performed. On current PC's, $wordsize = 64$. There are several reasons for focusing on operations in terms of words: memory contents are fetched in blocks that are a multiple of *wordsize* and the most basic operations like add and multiply on signed or unsigned integers operate on values w of *wordsize* length (in bits). An imperfect (it is architecture and implementation dependent) but useful intuition is that operations like adding two *wordsize* bit integers should take "one cycle" when the integers are in exactly the right place (registers, not memory, which takes hundreds of cycles to get to on a cache miss). Multiplying two integers by contrast can take $5-12$ cycles even if they are in the right place. As a practical matter, we usually do arithmetic on words, not bits. Words, w, of *wordsize* length can represent unsigned integers satisfying $0 \leq w \leq 2^{wordsize} - 1$ or signed values satisfying $-2^{wordsize-1} \leq w \leq 2^{wordsize-1} - 1$. From the point of view of bit based input arguments using primitive binary operations (and, or, xor), adding an m bit number and n bit number takes $O(max(m, n))$ time and $O(m + n)$ space, while multiplying an m bit number and n bit number takes $O(mn)$ time and $O(m + n)$ space.

There are a number of cryptographic attacks where the size of the system we can reasonable expect to attack is related to critical algorithms. We mention three important such algorithms (1) the extended Euclidean Algorithm which calculates the greatest common denominator $g = (a, b) = au + bv$ of two integers; (2) The LU algorithm which solves n linear equations in n unknowns and, (3) The FFT algorithm which evaluates or interpolates (finds the coefficients of the polynomial given values) polynomials.

The Euclidean Algorithm is arguably the most important "non-trival" algorithm in computer science. It is based on the division algorithm. Here it is in slightly telegraphic form: Suppose $a > b, a, b \in \mathbb{Z}$. Using the division algorithm, we can construct the following sequence mechanically: $a = bq_1 + r_1$, $b = r_1q_2 + r_2$, $r_1 = r_2q_3 + r_3$, ..., $r_{k-2} = r_{k-1}q_k + r_k$, $r_{k-1} = r_kq_{k+1}$. Now $r_1 = a - bq_1$ and $r_2 = b - r_1q_2$ imply that $r_2 = b - q_2(a - q_1b) = b(q_1q_2 + 1) - q_2a$. Continuing this thru the penultimate step of the sequence, it is easy to see that $g = r_k = au + bv$ for integers u and v that are obtained by following the procedure above. This is the extended Euclidean Algorithm (extended, because we produced u and v). g is the greatest common denominator (the British call it the highest common factor or HCF which is a much better name). The number of equations in this sequence is roughly $lg(b)$ ($lg(x) = log_2(x)$). So this procedure is very efficient. The same thing works for univariate polynomials. This form, $g = au + bv$, of the HCF is called the Plucker or Bezout form and it is very, very useful. From it, we can pick off the inverse of $a \pmod b$ when $g = 1$. As we saw in the proof of the Chinese Remainder Theorem, u, v are important in obtaining solutions to simultaneous congruences in co-prime moduli.

Finding the extended gcd of an m bit number and n bit number takes $O(mn)$ time and $O(m + n)$ space. Finding (a, b) has average running time: $O((1 + \frac{max(u,v)}{(u,v)})lg(min(u, v)))$.

On to solving linear equations. Suppose we want to find solutions to the following set of equations:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n} = c_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n} = c_2$$

$$\ldots$$

$$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = c_n$$

We apply a simple but beautiful procedure, called Gaussian elimination, which proceeds as follows: (1) Suppose $a_{11} \neq 0$ (if not, rearrange the equations so that this is true [6]); (2) Divide the first equation by a_{11}; (3) Subtract a_{k1} times equation 1 from equation k for $k = 2, 3, \ldots, n$. This yield an equivalent system of equations:

$$x_1 + a'_{12}x_2 + \ldots + a'_{1n} = c'_1$$
$$0 + a'_{22}x_2 + \ldots + a'_{2n} = c'_2$$
$$\ldots$$
$$0 + a'_{n2}x_2 + \ldots + a'_{nn}x_n = c'_n$$

This last $n-1$ equations are linear in $n-1$ variables and we can repeat the procedure to get and equivalent set of equations:

$$x_1 + a'_{12}x_2 + \ldots + a'_{1n} = c'_1$$
$$0 + a'_{22}x_2 + \ldots + a'_{2n} = c'_2$$
$$0 + 0 + a''_{13}x_3 + \ldots + a''_{1n} = c''_3$$
$$\ldots$$
$$0 + 0 + \ldots + 0 + a''_{nn}x_n = c''_n$$

If you write this out in matrix notation the coefficient will be upper triangular if the matricies operate from the right and lower triangular if they operate from the left. Now we can follow a similar procedure from the bottom up to remove all non-zero coefficients of x_n from all but the last equation, then all non-zero coefficients of x_{n-1} from all but the second to last equation and so on yielding an equivalent system:

$$b_{11}x_1 + 0 + \ldots 0 = d_1$$
$$0 + b_{22}x_2 + 0 + \ldots + 0 = d_2$$
$$\ldots$$
$$0 + 0 + \ldots + b_{nn}x_n = d_n$$

which we can easily solve. Notice that the matrix for this equivalent system is diagonal matrix

$$diag(b_{11}, b_{22}, \ldots, b_{nn}) = (d_1, d_2, \ldots, d_n).$$

A slightly stuffier sounding but equivalent procedure from the point of view of matrix based algorithms is called LU-factorization. The principle result (which is a consequence of applying Gaussian elimination is: Let $A \neq 0$ be an $m \times n$ matrix. There are permutation matricies P, Q such that $P^T AQ = LU$ where L is lower triangular and U is upper triangular. If multiplying two $n \times n$ matricies takes $T_M(n)$ time and $T_{LU}(n)$ represents the time to perform LU, $T_{LU}(m) \leq mCT_M(m)$ for some constant C. Some sparse systems can be solved much more quickly because multiplying sparse matricies can take linear time if we're very lucky. In describing Gaussian elimination and LU-factorization, we assumed "infinite precision" arithmetic. When the matricies consist of floating point numbers on real computers, errors and error propagation becomes the major implementation problem. Fortunately, in cryptography, we often are solving equations over finite fields where there is no round-off error in sight. A couple of other matrix algorithms come up, from time to time, in cryptanalysis. QR-factorization via unitary operations is used in the least square approximation problem. Spectral decomposition with hermetian conjugates: $U^H AU = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is used in Principal Component Analysis.

The Discrete Fourier Transform of a vector $\vec{a}$ is obtained by calculating $F(\vec{a}) = A\vec{a}$ where $A = \omega^{ij}$ and ω is a primitive n-th root of unity. Notice this corresponds to evaluating the polynomial $\vec{a} \cdot (1, x, x^2, \ldots, x^n)$ with x taking on the value of each of the n-th root of unity. The inverse of A exists (A is just a Vandermonde matrix) and, in fact, $A^{-1} = \frac{1}{n}\omega^{-ij}$. The inverse operation is really just the interpolation of the

[6] If you can't rearrange the equations to do this, x_1 is inessential and you really have n equations in $n-1$ unknowns.

coefficients of the polynomial from its values on the roots of unity. Doing the matrix calculation can obviously be done in no more than $O(n^2)$ multiplications but Tukey and Cooley noticed that, for this special matrix, the calculation could be performed in $O(nlg(n))$ time. They reasoned as follows: Let $n = pq$ and set $j = j(j_1, j_2) = j_1 q + j_2, k = k(k_1, k_2) = k_2 p + k_1, 0 \leq j_1 < p, 0 \leq j_2 < q, 0 \leq k_1 < p, 0 \leq k_2 < q$. Then $\hat{f}(k_1, k_2) = \sum_{j_2=0}^{q-1} e^{\frac{2\pi i j_2 (k_2 p + k_1)}{n}} \sum_{j_1=0}^{p-1} e^{\frac{2\pi i j_1 k_1}{p}} f(j_1, j_2)$. This requires $p^2 q$ and $q^2 p$ operations respectively or $pq(p+q)$ rather than $(pq)^2$. Applying this recursively, if n is a power of 2, the produces the desired result.

There are some other computer science problems where a detailed study of algorithms and performance of those algorithms are very important in cryptanalysis: (1) sorting n quantities (say integers), (2) finding a shortest path, (3) finding an assigment of truth values to literals that make a specified set of clauses all true or showing that no such assignment exists (SAT), (4) solving a set of non-linear equations over a finite field (Non-linear Solve).

Sorting, that is, putting n objects in order, is $\Theta(nlg(n))$. We can show sorting is $O(nlg(n))$ by applying mergesort and demonstrating that performance is dictated by the recurrence $T(n) = 2T(\frac{n}{2}) + (n-1)$. The first term is the cost of applying mergesort to each half of the sort and the $n-1$ comes from the merge. One can also show that sorting with comparisons can't be faster in general than $nlg(n)$; here's how: There are $n!$ possible arangements of the objects to be sorted and all arrangements are possible in a general instance. Any sorting algorithm must have enough comparisons to distinguish among the $n!$ possibilities and, at best, any binary comparison distinguishes between two balenced sets of possibilities. Thus, there must be $lg(n!)$ comparisons. $lg(n!) \approx lg(\sqrt{2\pi n}(\frac{n}{e})^n) \geq nlg(n)$. Of course, this latter argument works applies only to the general sorting problem using comparisons. On restricted problems, we can often do better: bucket-sort is $O(n)$ but is restricted to sorting a fixed (usually small) number of values and topological sort for putting a partially ordered set in any order consistant with the order relation is also $O(n)$.

Finding the shortest path in a graph takes $O(n^2)$ time.

The remaining two problems are more interesting. They fall into the catagory of NP-complete ("NPC") problems. NPC problems are those where verifying a solution takes polynomial time in the input size but for which all known algorithms that solve a general instance seem (but have not been proved) to take exponential time. All NPC problems can be transformed to each other by a polynomial transformation so if a polynomial-time algorithm exists for any NPC problem, any NPC problem has a polynimial time algorithm. Let P be the class of problems that are polynomial time and N be the class of problems that are "non-deterministic polynomial time" (where a solution can be verified in polynomial time).

$P \subseteq N$. If $A \leq B$ [7] and $B \in P$ then $A \in P$. $L \in NPC$ if and only if $L \in NP$, $A \in NP \rightarrow A \leq L$. Classical computation theory classifies problems by a "certain" solution on all instances. Later we will encounter problems which can be solved in polynomial time "up to an arbitrary error, ϵ" and call the class RP for "randomized polynomial." $P \subseteq RP \subseteq NP$. There are actually two ways in which the solution can be subject to error. In "Monte-Carlo" algorithms, the problems are always solved by a PTM (probabalistic Turing machine) and the answers given are the machine are "wrong" no more than ϵ of the time. In "Las Vegas" algorithms, the answer given by the PTM is always right but the PTM may return "I don't know" ϵ of the time. As an example, $COMPOSITE$ which answers the question "is n composite" is in RP. The naive factoring algortihm is $O(\sqrt{n})$ time, and, we can discover whether a number is prime as an outcome of this very expensive procedure [8] but there are probabalistic algorithms which, while not polynomial, are sub-exponential. Primality testing (answering $COMPOSITE$ directly) with probabalistic algorithms (using the Miller-Rabin test, for example) is very efficient. This is a very important application of a PTM. The big questions for us are: what about SAT and Discrete log?

Sastifiability or SAT is the prototypical NP-complete problem. The input is a number, $O(n)$, of literals or variables and a number, again $O(n)$, of clauses like $x \vee y \vee z$. A solution consists of a variable assignment to true or false which simultaneously makes all the clauses true. Verifying that a given proposed variable assignment simultaneously satisfies *all* the clauses takes linear time but no one knows a general algorithm

[7] $A \leq B$ means problem A can be transformed to problem B in polynomial time; this is called a reduction *from A to B*.

[8] Remember n is the number to factor, the bit representation length, b, is logrithmatic in n, so $\sqrt{n} \approx 2^{\frac{b}{2}}$. Thus this algorithm is exponential in the input length.

to find such an assignment in polynomial time for an arbitrary set of clauses. Solving, nonlinear polynomial equations called "Non-linear Solve" can be described as follows: Given an $O(n)$ list of variables and an $O(n)$ list of equations find the an assignment of values to the variables (in a finite field) that satisfies all equations simultaneously. Even if the equations are restricted to those that are quadratic, at worst, this problem is in NPC. Non-linear Solve also appears to be asymptotically inaccessible in the general case. If it were easy to solve these equations, according to Shannon's dictum, cryptanalysis would generally be easy.

SAT and Non-linear Solve share a very interesting property. A SAT problem instance in which the number of clauses is small compared to the number of variables is often quite easy to solve and it's often very easy to show there is no satisfying assignment when there are a lot of clauses compared to the number of variables in an instance. In fact, SAT problems turn from very easy to very hard abruptly when the ratio of the number of clauses to the number of variable approache 4.3. The same phenomena occurs in Non-linear Solve. So cryptographers are motivated to find transformations that map complicated (i.e.- well balenced) looking equations into a very sparse one. In addition, there is a very practical way to map Non-linear Solve over $GF(2)$ to a SAT problem.

10 References

Whew! I hope you weren't discouraged.

Many of the algebra texts mentioned in the references on the main web site are very nice and cover all this material (and much more) in an instructive and entertaining way. Two of my favorites are Herstein and van der Waerden. For algorithms, Aho et. al. and Corman, et. al. are good. For linear algebra, Hoffman and Kunze or the new book by Leibler are nice. On the other hand, doing is more important than reading.

Please send corrections to jmanfer@microsoft.com or, if I get fired from Microsoft, to JohnManferdelli@hotmail.com.