T-110.5211 Cryptosystems

Review of crypto mathematics
Classical ciphers

18.9.2008

Kaufman et al: Chapters 2.1 / 6.2 / 7.1-7.9 / 8.1-8.5
Stallings: Chapters 2.1-2.7 / 4.1-4.8 / 8.1-8.7

Outline

- Review of crypto mathematics
  - Number theory basics, modular arithmetic
  - Groups, rings, and fields
- Classical ciphers

Review of crypto mathematics

What sort of mathematics is used?

- Binary arithmetic (bit vectors, bit shifts, bit operations, ..)
  - Unkeyed primitives (e.g. unkeyed hash functions)
  - Symmetric and asymmetric primitives
- Finite groups and fields
  - Asymmetric primitives, some symmetric primitives (e.g. AES)
- One-way trapdoor functions
  - Asymmetric primitives
- Number theory (primality, etc)
  - Asymmetric primitives
- Complexity theory & computational number theory
  - For computational security
  - All primitives, but especially asymmetric
  - => How difficult is some particular sort of attack?

What we're going to review

- Number theory basics
  - Primes, co-primes, gcd, lcm (Euclidean algorithm)
  - Modular arithmetic (mod n), modular inverses and exponentiation
  - Euler's totient function, Euler's theorem, Fermat's theorem
  - Generators and primitive roots (elements)
  - Chinese Remainder Theorem (CRT)
  - Primality testing
- Groups, rings, fields
  - Groups
  - Rings and fields
  - Finite fields – **GF(p^n)**, example of a field used in AES

Divisibility, primes, co-primes, gcd, lcm

- Divisibility
  - **a** | **b** => exists integer **k** such that **b = ka**
  - Non-trivial divisor if **a** not **±1** or **±b**
  - Composites have non-trivial divisors, primes don't
  - Fundamental theory of arithmetic (unique factorization, primes)
- Greatest Common Divisor (GCD), **gcd(a,b)**
  - Greatest **c** such that **c** | **a** and **c** | **b**
  - Computing – Euclidean algorithm; prime factorization
- Least Common Multiple (LCM), **lcm(a,b)**
  - Smallest **c** such that **a** | **c** and **b** | **c**
  - gcd and lcm satisfy **ab = gcd(a,b) lcm(a,b)**
  - Computing – equivalent to gcd (solve from above)
- Co-prime (relatively prime)
  - **a** and **b** coprime <=> **gcd(a,b) = 1**

Modular arithmetic

- **a ≡ b (mod n)** (**a** and **b** *congruent* **mod n**)
  - Congruent if **n** | **a-b** <=> **a = b + kn**
  - **n** is called the modulus
- When computing **mod n**, elements are integer sets
  - **{a, a±n, a±2n, ...}** considered equivalent, for each **a=0,...,n-1**
  - Can be represented uniquely using representatives **0,...,n-1**
  - Addition and multiplication retain congruence
  - Almost ordinary arithmetic – e.g. **a^b + cd - 4 = ... (mod n)**
- In practice computations using integers in range **[0, n-1]**
  - Reduction: Add or subtract a multiple of **n** to get back into **[0, n-1]**
  - Example (Java): `t = (t % N);`
  - Reduce whenever possible to speed up computation

Modular arithmetic

- Inverse exists for element **a (mod n)** iff **gcd(a,n)=1**
  - For **n** prime, satisfied if **a** non-zero (and in range [0,n-1])
  - For **n** composite, some non-zero elements don't have inverses
- Extended euclidean algorithm for **gcd(a,n) = b**
  - Finds **s** and **t** such that **sa + tn = b**
  - => **sa = b (mod n)**
  - => If **gcd(a,n)=1**, finds **s** such that **sa≡as≡1 (mod n)**
  - Thus, **s** is **a**'s inverse
  - The algorithm itself – look up references
- Computation **mod p** (**p** prime) typical in cryptography
  - Modulus prime => **gcd(n,p)=1** for all **n=1,...,p-1**
  - Thus all non-zero elements have multiplicative inverses

Modular arithmetic examples **mod 7**

- Addition and subtraction
  - **3 + 6 ≡ 9 (mod 7) ≡ 2 (mod 7) ≡ -5 (mod 7)** (etc)
  - **3 – 6 ≡ -3 (mod 7) ≡ 4 (mod 7) ≡ 11 (mod 7)**
- Multiplication
  - **3 * 6 ≡ 18 (mod 7) ≡ 11 (mod 7) ≡ 4 (mod 7)**
- Division
  - **x / y ≡ x y^{-1}, y^{-1}** exists if **y != 0** (when modulus is prime)
- Inverse element
  - Zero has no inverse; others have an inverse because 7 is prime
  - E.g. let inverse of 3 be x (unknown), what could x be? (trial)
    - 3*1≡3 / 3*2≡6 / 3*3≡9≡2 / 3*4≡12≡5 / 3*5≡15≡1 (bingo)
  - => Inverse of 3 in our field is 5 => **3*5 ≡ 15 ≡ 8 ≡ 1 (mod 7)**
  - If modulus composite, some numbers have no inverse
  - Extended Euclidean is efficient for finding inverses

Modular (discrete) exponentiation

- Modular exponentiation is important: **$f(x,y) = x^y \pmod n$**
  - Commonly used in e.g. public-key cryptography
- Fast algorithm for exponentation: **square-and-multiply**
  - Factor **y** into binary form: **$y = y_1 + 2y_2 + 4y_3 + 8y_4 + \ldots$**
  - Write: **$x^y \equiv x\text{^}(y_1 + 2y_2 + 4y_3 + 8y_4 + \ldots) \equiv (x^1)^{y1} * (x^2)^{y2} * (x^4)^{y3} * \ldots$**
  - If **$y_i = 0$** for some term, the term becomes **1** and can be omitted
  - Algorithm: initialize result **r=1** and **t=x**; then loop **i=1,...,nbits(y)**
    - If bit **y_i=1** => **r := r * t** (where **$t=x^{2\text{^}(i-1)}$**) [reduce]
    - Square temporary: **t := t * t** [reduce]
    - Loop until no more binary digits in **y**, at end, **r** contains result
- The resulting algorithm is fast and in common use
  - Commonly available in crypto libraries (**modpow, modexp**, ...)
- But what about roots and logarithm ?

Modular (discrete) roots and logarithm

- Inverses of modular exponentiation **$x^y = z \pmod n$**
  - Not straightforward like exponentiation
- Discrete roots – **z** & **y** known, compute **x**
  - Complexity depends on **n**
  - RSA security is based on difficulty of discrete roots, when n is composite but factors not known
- Discrete logarithm – **z** & **x** known, compute **y**
  - Known as the Discrete Logarithm Problem (DLP)
  - For some groups believed to be very difficult, e.g. **mod p** (prime)
  - Diffie-Hellman key exchange is based on DLP
- Discrete vs. modular (terminology)
  - Discrete applies to all groups, not just modular arithmetic

Euler's totient, {Euler's,Fermat's} theorem

- Euler's totient function (phi function), **φ(n)** (**n ≥ 1**)
  - Number of integers in range **[1,n]** that are coprime to **n**
    - Examples: **φ(4) = 2**(*), **φ(7) = 6**, **φ(8) = 4**
  - Arithmetic rules
    - For primes and prime powers: **$\varphi(p) = p-1$**, **$\varphi(p^k) = (p-1)\, p^{k-1}$**
    - If **a** and **b** coprime => **φ(ab) = φ(a) φ(b)**
  - Any prime factorization **$n = p_1^{k1}\, p_2^{k2} \ldots p_r^{kr}$**
    - **=> $\varphi(n) = \varphi(p_1^{k1}) \ldots \varphi(p_r^{kr}) = (p_1-1)p^{k1-1} \ldots (p_r-1)p^{kr-1}$**
  - Importance – size of multiplicative subgroup **mod n**
- Fermat's (little) theorem for a prime **p**, **a ≥ 1**
  - **$a^p \equiv a \pmod p$** (**or** **$a^{p-1} \equiv 1 \pmod p$**)
- Euler's theorem for integer **n ≥ 1** (generalizes Fermat)
  - **$a^{\varphi(n)} \equiv 1 \pmod n$**, **a ≥ 1** is coprime to **n** **(gcd(a,n)=1)**
  - Allows reductions: **$a^z \equiv a^{b+k\varphi(n)} \equiv a^b (a^{\varphi(n)})^k \equiv a^b\, 1^k \equiv a^b \pmod n$**

(*) Example: gcd(1,4)=1, gcd(2,4)=2, gcd(3,4)=1, gcd(4,4)=4 => φ(4) = 2

Primitive roots (elements)

- Pick a number and multiply it by itself repetitively
  - **g, g^2, g^3, g^4**, ... (all **mod n**), eventually repeats
  - Thus **g** generates a certain set (group) of numbers, **<g>**
- Primitive element (of a group, **G**)
  - A generator **g** is a primitive element of a group G if **<g> = G**
    - That is, every element of **G** can be expressed as a power of **g**
    - => Exponents of **g** can represent elements of G
  - Example **mod 7** where G={1,2,3,4,5,6} (+ multiplication)
    - 1 is not a p.e.: <1> = {1, 1, 1, ...} = {1}
    - 2 is not a p.e.: <2> = {2, 4, 1, 2, 4, 1, ...} = {1,2,4}
    - **3 is a p.e.**: <3> = {3, 2, 6, 4, 5, 1, 3, 2, ...} = {1,2,3,4,5,6} = G
    - 4 is not a p.e.: <4> = {4, 2, 1, 4, 2, 1, ...} = {1,2,4}
    - **5 is a p.e.**: <5> = {5, 4, 6, 2, 3, 1, 5, 4, ...} = {1,2,3,4,5,6} = G
    - 6 is not a p.e.: <6> = {6, 1, 6, 1, ...} = {1,6}
  - Finding primitive elements is not trivial, no easy formula known
  - For modular arithmetic the term primitive root is used

Chinese Remainder Theorem (CRT)

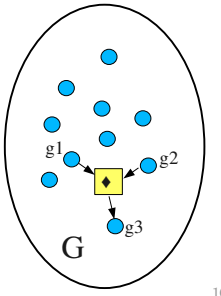
- Solving an equation such as **x ≡ 5 (mod 7)** is trivial
  - Solution: {5, 12, -2, 19, -9, ... }
- But what if **x** needs to satisfy a set of equations ?
  - **x ≡ 5 (mod 7)** **$x \equiv a_1 \pmod{n_1}$**
  - **x ≡ 3 (mod 8)** **$x \equiv a_2 \pmod{n_2}$**
  - **x ≡ 2 (mod 15)** **$x \equiv a_3 \pmod{n_3}$**
- Chinese Remainder Theorem (CRT)
  - **<u>If:</u>** all $\mathbf{n_i}$ are pairwise coprime (example above => OK)
  - **<u>Then:</u>** there is a unique solution (modulo $\mathbf{N = n_1\, n_2 \ldots n_k}$) for any $\mathbf{a_i}$
  - Example applications
    - Speed up computation, e.g. RSA
- Algorithm – check literature
  - Based on gcd & (extended) Euclidean algorithm

Primality testing

- Important to cryptographic security
  - Primality testing critical to security of e.g. RSA (keypair gen.)
  - Because large numbers are involved, performance is important
- Probabilistic algorithms
  - Try to prove that a claimed number is, in fact, a composite
    - If we fail to do so, the number is a *probable prime*
  - Currently the most practical algorithms – several algorithms exist
    - Rabin-Miller strong pseudoprime test, polynomial time
    - Solovay-Strassen test, etc.
- Deterministic algorithms
  - In general, much slower than probabilistic algorithms
  - Elliptic curve primality proving (ECPP)
  - Agrawal et al (2002) – "AKS" test, proven polynomial time
- We skip the algorithms in this course, use references

Mathematical group, **(G,♦)**

- A set of elements **G**, with an operation ♦, $(g_1 \diamond g_2) \rightarrow g_3$
  - The operation ♦ is simply a function with infix syntax
- ... satisfying the following conditions
  - **Closed**: ♦ always maps to another member of the set G
  - **Associative**: **a♦(b♦c) = (a♦b)♦c**, for all **a,b,c**
  - **Identity**: exists element **1** in **G**, such that **a♦1=1♦a=a** for all **a**
  - **Inverse**: for each **a**, there exists $\mathbf{a^{-1}}$ such that $\mathbf{a \diamond a^{-1} = a^{-1} \diamond a = 1}$
- A group G is abelian (commutative) if also
  - **Commutative**: **a♦b=b♦a** for all **a, b ϵ G**



Examples of groups

- A few definitions
  - $\mathbf{Z_n = \{0,1,...,n-1\}}$
  - $\mathbf{Z_n^* = \{a \epsilon Z_n \mid gcd(a,n) = 1\}}$ (multiplicative subgroup of $\mathbf{Z_n}$)
  - $\mathbf{Z_p^* = \{1, ..., p-1\}}$ where **p** is prime
- Examples
  - $\mathbf{(\{0,1,2,3\}, f(g_1, g_2) = (g_1+g_2)\%4)}$ (commutative)
  - $\mathbf{(Z_p^*, * \ (mod\ p))}$ (commutative)
  - **(M, x)** where **M** is set of non-singular square matrices, **x** is matrix multiplication (noncommutative)

Rings and fields

- Ring **(R, +, *)** = set **R** with two binary operations (#)
  - **(R, +)** is an abelian (commutative) group with identity denoted **0**
  - **(R, *) is a** *monoid*(*) with identity denoted **1**
  - Identity elements of **+** and * are distinct: **0 != 1** (#)
  - Distributive: **a*(b+c) = (a*b) + (a*c) and (b+c)*a = (b*a) + (c*a)**
  - Commutative ring (in *): **a*b = b*a**
- Field **(F, +, *)**
  - A commutative ring in which all non-**0** elements have multiplicative inverses
  - In other words
    - **(R, +)** and **(R\{0}, *)** are abelian (commutative) groups
    - Distributive law (above)
    - Identity elements (for **+**, *) distinct: **0 != 1**

(*) (R, *) is a *monoid, i.e. fulfills all group properties except existence of inverses*
(#) Some variation exists for the definition of a ring (e.g. identity for *, associativity for *, 0 != 1)

Examples of fields

- $\mathbf{(Z_n, +, *)}$ is a field if and only if n is prime $\mathbf{(Z_p)}$
  - Inverses exist for all **{1...n-1}** if and only if n is prime
  - Usual definition for addition and multiplication (mod n)
- Examples unrelated to cryptography
  - Rational, real, and complex numbers
  - Integers with integer + and * are **not** a field, because multiplicative inverses only exist for **1** (= **1**) and **-1** (= **-1**)

"All finite fields are known" => $\mathbf{GF(p^n)}$

- Finite sets which satisfy field axioms are known
  - Called Galois fields (GF) in honor of Evariste Galois
- General finite field, $\mathbf{GF(p^n)}$ (**p** prime, **n** ≥ 1)
  - If $\mathbf{GF(p^1) = GF(p)}$ => the field is (isomorphic to) $\mathbf{(Z_p, +, *)}$
  - Elements of the field can be represented as symbolic polynomials
    - **n** terms with symbolic variable **x** in powers $\mathbf{x^0, ..., x^{n-1}}$
    - Coefficients of the powers are elements of $\mathbf{GF(p) = Z_p}$
  - Equivalence classes of polynomials
    - Relative to a *irreducible polynomial*, similar to modular arithmetic
    - Irreducible polynomial cannot be represented as a product of smaller polynomials (non-constant)
  - Arithmetic with elements = polynomial arithmetic + reduction
    - Addition for each coefficient separately (in **GF(p)**)
    - Multiplication using polynomial multiplication + polynomial reduction
- Importance – all other <u>finite</u> fields are isomorphic !

Example – **GF(2^8)** (used in AES)

- Elements of the field
  - Polynomials (deg 7) with binary coefficients from $\mathbf{Z_2 = \{0,1\}}$
  - Addition and multiplication in binary, denoted $+'$ and $*'$
  - Example: $\mathbf{1x^7 + 0x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 0x^1 + 1}$
  - Can also be thought of as 8-bit binary values: **10011001**
- Irreducible polynomial
  - AES uses $\mathbf{x^8 + x^4 + x^3 + x + 1}$ **(100011011)**
  - All other finite fields with $2^8 = 256$ elements are isomorphic
- Reduction modulo irreducible polynomial (remainder)
  - $\mathbf{x^8 + x^4 + x^3 + x + 1 \equiv 0 \pmod{x^8 + x^4 + x^3 + x + 1}}$
  - => $\mathbf{x^8 \equiv x^4 + x^3 + x + 1 \pmod{x^8 + x^4 + x^3 + x + 1}}$
    - Note: sign doesn't matter, because $\mathbf{-x = x}$ **in** $\mathbf{Z_2}$
    - Thus we can just replace $\mathbf{x^8}$ with $\mathbf{x^4 + x^3 + x + 1}$
  - => Any power of **x** can be brought down to range [0,7]

Arithmetic example in AES's **GF(2^8)**

- Addition (and subtraction) is basically binary XOR
  - $\mathbf{a = 1x^7 + 0x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 0x^1 + 1}$ **(10011001)**
  - $\mathbf{b = 0x^7 + 0x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 0x^1 + 0}$ **(00011000)**
  - $\mathbf{a + b = (1+'0)x^7 + (0+'0)x^6 + \ldots + (0+'0)x^1 + (0+'1)}$
    $\mathbf{= 1x^7 + 0x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 0x^1 + 1}$ **(10000001)**
  - Coefficient addition in $\mathbf{Z_2}$ (denoted $+'$)
- Multiplication using polynomial arithmetic
  - $\mathbf{a = x^5 + x^2 + 1}$ **(00100101)**
  - $\mathbf{b = x^6 + x^3}$ **(01001000)**
  - $\mathbf{a * b = (x^5 + x^2 + 1)(x^6 + x^3) = x^{11} + \underline{x^8} + \underline{x^8} + x^5 + x^6 + x^3 =}$
    $\mathbf{= x^{11} + x^6 + x^5 + x^3 = (x^8 * x^3) + x^6 + x^5 + x^3 =}$
    $\mathbf{\equiv (x^4 + x^3 + x + 1)*x^3 + x^6 + x^5 + x^3 = \ldots =}$
    $\mathbf{= x^7 + x^5 + x^4}$ **(10110000)**
  - Reduction – replace $\mathbf{x^8}$ with $\mathbf{x^4 + x^3 + x + 1}$
- In practice – using processor's **xor** & mult. lookup table

Classical ciphers

Enigma – 1910s

Enigma – from 1910s through WW2

Originally patented by Scherbius in 1918, for commercial use. Adopted and strengthened by German military before and during WW2.

Enigma is electromechanical, and weak by modern standards. During WW2 considerable effort was spent in cryptanalyzing Enigma.

Enigma's is a *rotor machine*; its encryption is based on *rotors*, wheels of letters which rotate during encryption.

Photo: http://www.w1tp.com/enigma Museum

Classical ciphers

- Developed before computers
  - Computation manually or using a physical device (e.g. Enigma)
  - Security is low by modern standards
- Relevance
  - Design of symmetric ciphers still has common elements
- Classical cipher messages consists of letters **A...Z**
  - No whitespace, numbers, etc – this is just for simplicity
  - Letters also equated with numbers **A=0**, ..., **Z=25**
- Encryption and decryption
  - Plaintext and ciphertext are strings of letters, key usually letter(s)
  - $\mathbf{E_K(p)}$ = encryption of plaintext **p** with key **K**
  - $\mathbf{D_K(c)}$ = decryption of ciphertext **c** with key **K**
- Let's look at basic classical ciphers and generalize

Classical ciphers – substitution

- Caesar cipher (shift cipher) – add key **K**
  - $\mathbf{E_K(p) = p + K \pmod{26}}$ $\mathbf{D_K(c) = c - K \pmod{26}}$
  - Example – key **K = 3** (“**D**”), plaintext “**ABC**” => ciphertext “**DEF**”
  - Weakness: frequency analysis of letters
    - E.g. in English, letter “**E**” is most frequent => find most frequent ciphertext letter (from lots of ciphertext), and solve for probable key
- Affine cipher – generalization of Caesar
  - Key **K=(a, b)**, where **a** coprime to **26** (= has inverse, $\mathbf{a^{-1}}$)
  - $\mathbf{E_K(p) = ap + b \pmod{26}}$ $\mathbf{D_K(c) = (c - b)\, a^{-1} \pmod{26}}$
  - Weakness: similar to Caesar, but need two letters for two equations, then solve **a** and **b**
- Substitution cipher – generalization of affine cipher
  - Key **K** is a permutation function from $\mathbf{Z_{26} \rightarrow Z_{26}}$
  - Weakness: similar to affine, but each letter solved separately

Classical ciphers – permutation

- Permutation cipher
  - Take blocks of **n** letters, and permute their order, block by block
    - Example: “ABCDE” => “DACEB”
  - The larger block size (**n**), the better the cipher
  - Plaintext needs to be padded to a full block length first
- Weaknesses
  - Original letters stay the same, just change position
  - Thus, frequency of letters is same as in plaintext
  - Block size can often be deduced because ciphertexts are all multiples of block size

Classical ciphers – alternating ciphers

- Polyalphabetic cipher (e.g. Enigma)
  - Substitution cipher with multiple permutation functions
  - How permutation function is changed depends on the cipher
- Example – Vigenère
  - Basically multiple Caesar ciphers, used in sequence
  - Key is a string, e.g. “CRYPTO”, which defines six Caesar ciphers
    - For first cipher, key K = 2 (“C”), for second K = 18 (“R”), etc.
  - Example of encryption
    - Plaintext: `ATTACKATDAWN`
    - Keystream: `CRYPTOCRYPTO`
    - Plaintext: `CKRPVYCKBPPB`
  - Weaknesses
    - If key length known, separate frequency analysis for each key letter
    - Key length can be determined using e.g. Kasiski test (see refs)
    - Repeated ciphertext sequences may be caused by repeating plaintext & keystream

Classical ciphers – miscellaneous

- Running key cipher
  - Sender and receiver have the exactly same book, word for word
  - Sender indicates an offset to the book text, e.g. page, row, word
  - Letters from the book starting from the offset are taken as the keystream
  - Plaintext is Caesar-encrypted, key **K** from keystream
- Hill cipher
  - Blocks of **n** plaintext letters thought of as **n**-dimensional vectors
  - Key **K = (A, b)**, **A** invertible **n*n** matrix
  - $\mathbf{E_K(p) = A*p + b}$ (matrix multiplication, field $\mathbf{Z_{26}}$)
  - $\mathbf{D_K(c) = A^{-1} * (c - b)}$ (matrix multiplication, field $\mathbf{Z_{26}}$)
  - Combines substitution and permutation using a linear transform
    - Can also be thought of as a substitution cipher, if alphabet is assumed to consist of n-letter words

Classical ciphers – one-time pad

- One-time pad
  - Like running cipher, but
    - (a) keystream is fully random
    - (b) keystream is never repeated (even in other messages)
  - One-time pad is information-theoretically secure
    - Shannon showed that knowledge of ciphertext gives no information about plaintext (= perfect secrecy)
    - To simplify: one decryption is as plausible as the next
  - Drawback: keystream is as long as the message
    - => How to transfer keystream ? (advantage: before message...)
- One-time pad has relevance even today
  - Stream ciphers are essentially one-time pads, where random keystream is replaced with a pseudorandom keystream
  - Pseudorandom keystream is computationally difficult to predict (cryptographic PRNG)
  - Cold War “Hot line”, “number stations” (possibly)(*)

(*) Google for “Conet Project”

Relevance of classical ciphers today

- One-time pad is still an important construct
  - Ideal encryption algorithm – secure and simple
- Algorithm is usually public, but key is kept secret
  - Kerckhoffs' principle – a cryptosystem should be secure if the algorithm is known but key is unknown
- Substitution and permutation
  - Substitution and permutation with sufficient iteration are still used for block cipher design
  - **Product cipher** – multiple simple ciphers in sequence
  - Modern ciphers operate on bits and bytes

Summary

- Review of crypto mathematics
  - Number theory basics, modular arithmetic
  - Groups, rings, and fields
- Classical ciphers