Cryptanalysis

Introduction to Public Key Systems: RSA

John Manferdelli
JohnManferdelli@hotmail.com

Public Key (Asymmetric) Cryptosystems

- An asymmetric cipher is a pair of key dependant maps, (E(PK,-), D(pK,-)), based on related keys (PK, pK).
- D(pK, E(PK,x))=x, for all x.
- PK is called the public key. pK is called the private key.
- Given PK it is infeasible to compute pK and infeasible to compute x given y= E(PK, x).

Diffie, Hellman, Ellis, Cocks, Williamson. Diffie and Hellman, "New Directions in Cryptography", IEEE Trans on IT 11/1976. CESG work in 1/70-74.

Uses of Public-Key Ciphers

- Symmetric Key Distribution
- Key Exchange and other protocols
- Digital Signatures
- Sealing Symmetric Keys (SMIME)
- Authentication
- Proving Knowledge without disclosing secrets (used in anonymous authentication)
- Symmetric Key systems cannot do any of these. However, symmetric key systems are *much* faster than Public Key systems.

Symmetric Key Distribution

- Imagine you are the head of security for Microsoft and insist that all Microsoft financial communications transmitted over the Internet be encrypted for your finance machines. You buy "black boxes" for every Internet egress point, each with a known Public Key (the private key is generated on the black box and is known only to that hardware).
- Every day, just before 0^h Zulu, you generate a new symmetric key K_d, encrypt it and transmit $E(PK_i, K_d)$ to each black box, i, (Hopefully, using a mechanism that ensures that it comes from you or what happens?)
- What's good about this? Keys are never touched by humans.
- What would you do if you were worried that some black boxes could be compromised (i.e.- private keys determined)?

Digital Signatures

- I want to send you messages you can rely from time to time, like:
  - M="I, John Manferdelli, promise on March 1, 2013, that (1) I will give everyone in CS 294-90 an A, (2) I will eat my vegetables."
- How can I prove (electronically) they come from me?
  - I generate a public/private key pair PK_{JLM}, pK_{JLM}.
  - One day in class I personally give you PK_{JLM} on a piece of paper.
  - When I send a message like M I also transmit: D(pK_{JLM}, SHA-256(M)).
  - When you get M, you calculate SHA-256(M) and compare it to E(PK_{JLM}, SHA-256(M)), if it matches, you can tell it's from me.

Sealing Symmetric Keys
(Key encapsulation)

- I want to send you a confidential document, M (like an email). I know your public key PK_{you} (maybe you told it to me, maybe it's in a directory, maybe someone I trust gave it to me and vouched for it).
- I generate a new symmetric key, K, at random.
- I encrypt M with CBC-AES using K and transmit to you:
  1. IV
  2. CBC-AES_K(IV,M)
  3. E(PK_{you}, K)
  4. I may also sign the message so you can be sure it came from me
- This is essentially how S/MIME mail works.

Authentication

- I am on a physically secure line (no one can eavesdrop or modify messages between me and the other end point) so I'm not worried about confidentiality.
- I want to make sure you, my lawyer, is on the other end and I know your public key PK_{you}.
- Before I say anything I'd regret reading in the New York Times, I generate a (big) random number, N and append the date and time, calling this entire message, M. I transmit M and ask you to apply $D(pK_{you}, M)$. If $E(PK_{you}, M)=M$, I can be sure it's my attorney; otherwise, I take the fifth.

Trapdoor functions

- Public Key systems are based on “computationally hard” “trap door problems”.
  - Find a function y=f(x) which is easy to compute but hard to invert without “secret” information, t forming the trapdoor. With t, it is easy to compute g(t, f(x))=x
- Some trapdoors
  - Factoring: $f(x) = x^e \ (mod\ n)$. $n = pq, p, q$ are “large” primes – thousands of bits long. Knowledge of p and q is the trapdoor.
  - Discrete Log: $h = g^a \ (mod\ p)$. $f(x) = (xh^b, gb)$, b selected at random by encryptor. Knowledge of a is the trapdoor.
  - Elliptic curve discrete log: Given Elliptic curve $E_p(a, b)$: $y^2 = x^3 + ax + b$, over a finite field Z_p, and a base point $B = nP$. Message is encoded as point on $E_p(a, b)$. $f(M) = (M + rB, rP)$. r is picked randomly by encryptor. Trapdoor is n.

Existing Public-Key Ciphers

- Rivest Shamir Andelman (1978)
  - Based on factoring
- El Gamal (1984)
  - Based on discrete log
- Elliptic Curve (1985, Miller-Koblitz)
  - Based on elliptic curve discrete log (over finite fields).
- New NIST PK systems
  - Based on shortest vector in lattice

Math for RSA

Some Number Theory

- Fundamental Theorem of Arithmetic
- Euclidean algorithm for GCD
- Solving congruences
- Chinese remainder theorem
- Integer arithmetic mod n
- Fermat's Theorem
- Quadratic Reciprocity: Legendre and Jacobi symbols.
- Primality testing

Fundamental Theorem of Arithmetic

- Let n be any positive integer, n can be written as a product of primes in an essentially unique way (except for units and the order of the primes).
- It may be hard to actually carry out this factorization.
- Easy to get rid of small factors.
- Suppose $n = ab$ and a as well as b are "large."
- Example: Factor 9,313,729 = $2713 \cdot 3433$
- Have to try factors until about $n^{1/2}$.
- Factoring is hard

Greatest common divisor

- $\gcd(a,b) = (a,b) = max_{t \in \mathbb{Z}^+}(t|a, t|b)$
  - (4,6)=2, (12, 36)=12, (2,5)=1
- Euclid's algorithm: If $a, b \in Z^+, a > b, \exists q, r \in Z^+ : a = bq + r, 0 \leq r < b$.
  - If $r = 0, b|a$.

- **Plucker**: $\exists x, y \in \mathbb{Z}$: $(a,b) = ax + by$
  Proof (and efficient computation) uses Euclidean algorithm. Suppose $a > b. a = bq + r$ and $t|a, t|b \rightarrow t|r$. Put $s = (a,b)$ and $a = r_0$ and $b = r_1$. We have $r_0 = q_1 r_1 + r_2$, and $s|r_2$. We compute successively $r_{i-1} = q r_i + r_{i+1}$, noting that $s|r_{i+1}$. Eventually, we come to a k: $r_{k-1} = q_k r_k$ We claim $(a,b) = r_k$. Observe that using these recursions, we can write $r_k = ax + by$; for example, the first two recursions give $r_2 = a - bq_1$ and $b = r_2 q_2 + r_3$, yielding $r_3 = b - r_2 q_2 = b - q_2(a - bq_1) = aq_2 + b(1 - q_1 q_2)$. It is also clear by induction that $r_k|s$ and $r_k|r_i, i < k+1$. Thus $r_k|a$ and $r_k|b$ so $s|r_k$. Since $r_k|s$ and $s|r_k$, $|s| = |r_k|$.

Chinese Remainder Theorem

- If $x = a_1 \ (mod\ m_1)$ and $x = a_2 \ (mod\ m_2)$ and (m_1, m_2)= 1, then the simultaneous equations have a unique solution mod m_1m_2.

- Proof: $\exists k_1, k_2 : k_1m_1 + k_2m_2 = 1$. Note $k_1m_1 = 1 \ (mod\ m_2)$ and $k_2m_2 = 1 \ (mod\ m_1)$. Set $a = a_2k_1m_1 + a_1k_2m_2$. This is a solution and a practical computational method.

CRT Example

- $N = 1517 = 37 \cdot 41$, solve $5x^2 = 2 \pmod{1517}$.
- First solve $5x^2 = 2 \pmod{37}$ and $5x^2 = 2 \pmod{41}$.
  - $(15)5 + (-2)37 = 1 \; and \; (-8)5 + (1)41 = 1$ so:
  - $x^2 = 2 \cdot 15 = 30 \; (mod \; 37)$ and $x^2 = 2x \; (-8) = 25 \; (mod \; 41)$.
  - $20^2 = 30 \pmod{37}$ and $5^2 = 25 \pmod{41}$.
- Use CRT
  - (10)37+(-9)41= 1
  - (5)(10)(37)+(20)(-9)(41)= 538= y (mod 1517)
  - $5(538)^2 = 2 \; (mod \; 1517)$.

Solving Congruences (mod n)

- Solve $ax = b\ (mod\ n)$
  - Same as for n=p, except some equations will not have solutions.
  - Procedure: Suppose, $(a, n) = 1$, then $au + nv = 1$ with $u, v\ \in \mathbb{Z}$. Put $x = ub$. $au = 1\ (mod\ n)$, so $aub = b\ (mod\ n)$. If $(a, n) \nmid b$, the equation has no solution. If $(a, n) | b$, solve $\frac{a}{(a,n)}x = \frac{b}{(a,n)}\ \left(mod\ \frac{n}{(a,n)}\right)$; this yields a solution.

Solving Congruence Example $mod\ n$

- Solve 3x=2 (mod 55)
  - 3(-18)+55(1)=1
  - 3(-18)=3(55-18)=3(37)=1 (mod 55)
  - 3(37x2)= 2 (mod 55)
  - So ,74= 19 (mod 55) is a solution: $3 \cdot 19 = 57 = 2\ (mod\ 55)$

A little group theory

- A group, G, is a set with an operation (usually written as multiplication) such that:
  - $a * (b * c) = (a * b) * c$
  - $a * 1 = 1 * a = a$
  - $\forall a \; \exists a^{-1}: a * a^{-1} = 1$
- S is a subgroup of G, if $S \subseteq G$ and S is a group.
- Lagrange's Theorem: if S is a subgroup of a finite group G, then $|S| \mid |G|$.

  Proof: Let $a \in G$ then $Sa = \{sa: s \in S\}$. Sa is called a coset. $Sa = Sb$ or $Sa \cap Sb = \emptyset$. If $s_1 a = s_2 b$, then ${s_2}^{-1} s_1 a = b$ and $Sb = S{s_2}^{-1} s_1 a =$ Sa. Thus, G is a its cosets.
- Example (cyclic groups):
  - $C_n = \{g^1, g^2, \ldots g^n = 1\}$. $g^i \; g^j = \; g^{(i+j)(mod\, n)}, g^0 = 1$.
- $C_6 = \{1, g, g^2, g^3, g^4, g^5\}$. $S_0 = \{1\}, S_1 = \{1, g^2 \; g^4\}$ and $S_2 = \{1, g^3\}$ are all subgroups of C_6. Note that $1|6$, $3|6$ and $2|6$.

Multiplicative group $mod\ p$

- $\mathbb{Z}_p^* = \mathbb{Z}_p\ - \{0\}$ is the multiplicative group $mod\ p$.
- $|\mathbb{Z}_p^*| = p - 1$
- So, if a $\in \mathbb{Z}_p^*$, $a^{p-1} = 1\ (mod\ p)$
- A theorem (we won't prove it) is that $\mathbb{Z}_p^*$ is a cyclic group. The generators are called primitive elements.
- Since $p - 1$ is even (unless p=2), $\mathbb{Z}_p^*$ always has a subgroup of order $\frac{p-1}{2}$.

Primitive roots $mod\ p$

- Wilson's theorem: $a^{p-1} = 1\ (mod\ p)$
  Proof: The size of the multiplicative group, Z_p^*, is $p-1$. $< a^i, i = 1, \ldots . >= S$ is a subgroup of the multiplicative group. $|S|$ = smallest $t \geqq 1$: $a^t = 1\ (mod\ p)$. By Lagrange, $t = |S|\ |\ |(p-1)$ so $a^{p-1} = 1\ (mod\ p)$.
- Generators of Z_p^* are also called primitive roots. Primitive roots are "irreducible" solutions to $x^{p-1} = 1$
- Example: p=11, 2 is a primitive root

i	2^i (mod 11)
1	2
2	4
3	8
4	5 (=16)
5	10

i	2^i (mod 11)
6	9 (=20)
7	7
8	3
8	6
10	1

Composite moduli and $\varphi(n)$

- $\mathbb{Z}_n^* = \{0 < x < n: (x, n) = 1\}$.
- $\mathbb{Z}_n^*$ is a multiplicative group:
  Proof: $x, y \in \mathbb{Z}_n^* \rightarrow (x, n) = (y, n) = 1$ so $(xy, n) = 1$ and $xy \in \mathbb{Z}_n^*$. If $ax = ay \ (mod\ n)$, and $a, x, y \in \mathbb{Z}_n^*$ then $x = y$ since $a(x - y) = kn$ and $(a, n) = 1 \rightarrow (x - y) | n \ so \ x = y \ (mod\ n)$. Now, since $\mathbb{Z}_n^*$ is finite, $x^i = x^j$ for some $i \neq j$. So, $i > j$, $x^{i-j} = 1 \ (mod\ n)$ by the cancellation property. $x^{i-j} = 1 \ (mod\ n)$ and x^{i-j-1} is the inverse of x.
- $\varphi(n) = |\mathbb{Z}_n^*|$.
- If $a \in \mathbb{Z}_n^*, a^{\varphi(n)} = 1 \ (mod\ n)$. For n=p, $\varphi(p) = p - 1$ and this is just Wilson's theorem.
- If $(a, b) = 1, \varphi(\mathrm{ab}) = \varphi(a)\varphi(b)$
  Proof: Since $(a, b) = 1, \exists x, y \in Z: ax + by = 1$. Suppose $(a_i, a) = 1$ and $(b_j, b) = 1$, put $t_{ij} = b_i ax + aiby$. $t_{ij} = ai \ mod\ a)$ and $t_{ij} = b_j \ (mod\ b)$. By the CRT, t_{ij} is unique $mod\ ab$ and further, $(t_{ij}, a) = 1 = (t_{ij}, b)$. There are $\varphi(a)\varphi(b)$ such t_{ij} so $|\mathbb{Z}_{ab}^*| = \varphi(a)\varphi(b)$.

The Multiplicative Group $(mod\ n)$

- $\varphi(p^e) = (p-1)p^{e-1}$
- If $n = \prod_{i=1}^{k} {p_i}^{e_i}$, $\varphi(\mathrm{n}) = \mathrm{n}\prod_{i=1}^{k}(1-\frac{1}{p_i})$

  Proof: $\varphi(n) = \varphi\left(\prod_{i=1}^{k} {p_i}^{e_i}\right) =$
  $\prod_{i=1}^{k} \varphi({p_i}^{e_i}) = \prod_{i=1}^{k} {p_i}^{e_i-1}(p_i-1)$
- $|\mathbb{Z}_n^*| = \varphi(n)$, so $(a,n) = 1 \rightarrow a^{\varphi(n)} = 1\ (mod\ n)$

  Proof: $|\mathbb{Z}_n^*| = \varphi(n)$ so by Lagrange's Theorem the size of the subgroup generated by a is the smallest integer $|a|$, such that $a^{|a|}=1$ divides $\varphi(n)$ so $\varphi(n) = |a|m$. As a result, $a^{\varphi(n)} = a^{m|a|} = 1$.

For n composite, multiplicative group need not be cyclic

- $a^{\varphi(n)} = 1\ (mod\ n)$, but there may be an $m|\varphi(n)$ with this property too.
- Thus, the multiplicative group may not be cyclic

- Example 1:
  - $n = 15 = 3 \cdot 5, \varphi(15) = \varphi(3)\varphi(5) = 2 \cdot 4 = 8$, but if $(a, 15) = 1, a^4 = 1\ (mod\ 15)$
  - Another example: $\varphi(2800) = 960$, but if $(a, 2800) = 1, a^{60} = 1\ (mod\ 2800)$

a	a^4 (mod 15)
2	1
4	1
7	1
8	1

a	a^4 (mod 15)
11	1
13	1
14	1
1	1

$\varphi(n)$

- Examples: $\varphi(1) = 1, \varphi(5) = 4, \varphi(25) = 20, \varphi(135) = 40.$
- Special case for RSA: $n = pq, \varphi(pq) = \varphi(p)\varphi(\mathrm{q}) = (\mathrm{p} - 1)(\mathrm{q} - 1).$
- Suppose $e = 5$ and $(e, (p-1)(q-1)) = 1$
- There are integers d, m: $ed + m\varphi(n) = 1$
- Put $x = a^e \ (mod\ n)$:
  - $x^d = a^{ed} \ (mod\ n) = a^{1-m\varphi(n)} \ (mod\ n) = a.$
- Example:
  - n= 13x17= 221. $\varphi(221) = 12 \times 16 = 192.$
  - $5(-115) + 192(3) = 1.\ d = -115 = 192 - 115 = 77\ (mod\ \varphi(n)).$
  - a= 53 so x= 53^5= 53^2 53^2 53 (mod 221)= 157x157x53= 66 (mod 221)
  - 66^{77}=66^{64} x 66^8 x 66^4 x 66^1 (mod 221).
  - 66^1(mod 221)= 66, 66^2(mod 221)= 157, 66^4(mod 221)= 118, 66^8(mod 221)= 1.
  - 66^{77}=66^{64}x66^8x66^4x66^1(mod 221)=1x1x118x66=7788-35(221)=53.

Quadratic Residues

- Which elements, x, of $\mathbb{Z}_p^*$ have square roots?
  - Equivalently, y **ε** $\mathbb{Z}_p^*$ such that $x=y^2$?
  - Suppose g is a generator, look at $\{g, g^2,\ldots,g^{(p-1)}\}$.
  - If $x \in \mathbb{Z}_p^*$ has a square root, y, then $x^{(p-1)/2}=y^{(p-1)}=1 \pmod p$.
- If $x^{\frac{p-1}{2}} \neq 1 \ (mod\ p)$then x does not have a square root.
- Since $x^{(p-1)}=1$, $x^{(p-1)/2}$ is 1 or -1 (mod p). This leads to:
  - If $x^{(p-1)/2} =1 \pmod p$ then x has a square root.
  - If $x^{(p-1)/2} =-1 \pmod p$ then x does not have a square root.
- Define $\left(\frac{a}{p}\right) = 1$ if a has a square root and $\left(\frac{a}{p}\right) = -1$ if not . This is called the *Legendre symbol*.
- $\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \pmod n$.
- If a and b both have square roots (mod p), so does ab. If neither a nor b has a square root ab has a square root. If only one of a, b has a square root then ab does not have a square root.
- $\left(\frac{ab}{p}\right) = (\frac{a}{p})(\frac{b}{p})$

Law of Quadratic Reciprocity

- If p and q are primes $\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{(p-1)(q-1)}{2}\frac{}{2}}$
- The proof uses this Lemma: For any odd prime, p, suppose $(a,p) = 1$, let $\mathcal{S} = \{t = ka, 1 \le k \le \frac{p-1}{2} : least\ positive\ residue\ of\ t\ (mod\ p) > \frac{p}{2}\}$ and $n = |\mathcal{S}|$ then $\left(\frac{a}{p}\right) = (-1)^n$.

  Proof of Lemma: Let $r_1, r_2, \dots, r_n$ be the elements of $\mathcal{S}$ and $s_1, s_2, \dots, s_k$ be the remaining elements t ∈ $\mathbb{Z}^*$: $t \le \frac{p-1}{2}$. $n + k = \frac{p-1}{2}$. The $p - r_i$ are distinct and $p - r_i \ne s_j$ for any j. So,$(p - r_1)(p - r_2)\dots(p - r_n)s_1 s_2 \dots s_k = \frac{p-1}{2}$ and thus $(-1)^n \cdot a \cdot (2a) \cdot \dots \cdot \frac{p-1}{2}a = 1 \cdot 2 \cdot \dots \cdot \frac{p-1}{2}\ (mod\ p)$. Thus$(-1)^n \cdot a^{p-1/2} = 1\ (mod\ p)$ and $a^{p-1/2} = (-1)^n\ (mod\ p)$.

Proof: Let $\mathcal{S} = \left\{(x,y):\ 1 \le x \le \frac{p-1}{2}, 1 \le y \le \frac{q-1}{2}\right\}$. $|\mathcal{S}| = \frac{(p-1)(q-1)}{4}$. Let $\mathcal{S}_1 = \{(x,y) \in \mathcal{S}: qx > py\}$, $\mathcal{S}_2 = \{(x,y) \in \mathcal{S}: qx < py\}$. $\mathcal{S}_1 = \left\{(x,y) \in \mathcal{S}: 1 \le x \le \frac{p-1}{2}, 1 \le y < \frac{qx}{p}\right\}$. $|\mathcal{S}_1| = \sum_{x=1}^{p-1/2} \llcorner \frac{qx}{p} \lrcorner$ and $|\mathcal{S}_2| = \sum_{y=1}^{q-1/2} \llcorner \frac{px}{q} \lrcorner$. $|\mathcal{S}_1| + |\mathcal{S}_2| = \frac{p-1}{2}\frac{q-1}{2}$ and the theorem is true by the Lemma.

Quadratic Reciprocity Example

	7	11	13	17	29	31
7		-1	-1	-1	1	1
11	1		-1	-1	-1	-1
13	-1	-1		1	1	-1
17	-1	-1	1		-1	-1
29	1	-1	1	-1		-1
31	-1	1	-1	-1	-1	

- Entry in row i, column j is $p[i]^{([j]-1)/2}$

- $(7/11)(11/7)=(-1)^{5x3}=-1$
- $(7/13)(13/7)=(-1)^{6x3}=1$
- $(7/17)(17/7)=(-1)^{8x3}=1$
- $(11/31)(31/11)=(-1)^{15x5}=-1$

Large Integer Computation

- Almost all public key algorithms are based on “hard” number theory problems over enormous (e.g.- 2048 bit) integers.
- We need to know how to do arithmetic on computers with huge numbers
  - Addition/subtraction
  - Multiplication
  - Modulus
  - Modular inverses
  - Exponentiation
  - Testing Primality
  - Factoring

Algorithm Timings

- Adding two m-bit numbers takes O(m) time.
- Multiplying two m-bit numbers takes $<O(m^2)$.
- Multiplying a 2m-bit number and reducing modulo and m-bit number takes $O(m^2)$.
- Computing (a, b) for a, b< n takes $O(\ln^2(n))$ time (i.e.- fast). This is Euclid's Algorithm and it started Knuth, Euclid and everyone else off on computational complexity. If n has m bits this is $O(m^2)$.
- Testing a number n for primality takes $O(n^{c\lg(\lg(n))})=O(2^{cm\lg(m)})$.
- Best known factoring: $O(n^{c(\lg(n)^{(1/3)}(\lg(\lg(n))^{(2/3)})})=O(2^{cm(m^{(1/3)}(\lg(m)^{(2/3)})})$. [a lot longer].

Primes are plentiful

- Euclid: There are infinitely many primes
- Prime Number Theorem: The number of primes, $\pi(n)$, less than or equal to n is asymptotically equal to $\frac{n}{\ln(n)}$.
  - Spookily accurate even for pretty small n.
  - First proof using complex analysis sketched by Riemann, finished by Hadamard and de la Vallee-Poussin. "Elementary" proof by Erdos and Selberg.
- Chebyshev's Theorem: For x>200, $c_1 \frac{x}{\ln(x)} < p(x) < c_2 \frac{x}{\ln(x)}$, $c_1 = \frac{2}{3}, c_2 = 1.7$
  Pretty easy to prove.
- Bertrand's Postulate: For all $n > 2$ there is a prime between n and $2n$

Prime Distribution Example

n	p(n)	n/ln(n)	p(n)/(n/ln(n))
10	4	4.34	.9217
50	14	12.78	1.10
100	25	21.71	1.15
500	95	80.46	1.17
1000	168	144.76	1.16
10^6	78498	72382	1.08
10^9	50847478	48254949	1.05

Factoring and exponents

- Suppose n is the product of two (possibly unknown) primes, p and q. If $\varphi(n)$ is known, we can calculate p and q.

  Proof: If p and q are known, $\varphi(n) = (p-1)(q-1)$.

  Set $f(x) = x^2 - Ax + n$ where $A = 1 - \varphi(n)$. p and q are the roots of $f(x) = 0$. We can solve this equation.

- Note: If $(e, \varphi(\mathrm{n})) = 1$, we can calculate d:

  $ed = 1\ (mod\ \varphi(\mathrm{n}))$. If we know p and q, this theorem tells us if we know a universal exponent, we can calculate d.

Universal Exponent Theorem

- Suppose $n = pq$. Given $r > 0, a^r = 1 \ (mod\ n)$, for all a: $(a, n) = 1$, we can factor n.
  - So, if we know $\varphi(pq)$, we can factor n.

- Method: Let $r = 2^k\ m$, odd. Put $b_0 = am \ (mod\ n)$ and $b_{i+1} = {b_i}^2 \ (mod\ n)$.
  1. If b_0= 1, pick another a.
  2. If b_{i+1}= 1 and b_i= -1, pick another a.
  3. If b_{i+1}= 1 and $b_i > 1, d = (b_i - 1, n)$ is a non-trivial factor of n.

Universal Exponent Example

- Let n= 1517. Note that $a^{1440}=1 \pmod{n}$.
  - 1440= $2^5(45)$, m= 45.
  - $b_0= 2^{45}=401 \pmod{1517}$
  - $b_1= 401^2 =1 \pmod{1517}$. Try again.
  - $b_0= 3^{45}=776 \pmod{1517}$
  - $b_1= 776^2=1444 \pmod{1517}$
  - $b_2= 1444^2 =778 \pmod{1517}$.
  - $b_3= 778^2 =1 \pmod{1517}$.
  - (778-1, 1517)= (777, 1517)=37.

Representing Large Integers

- Numbers are represented in base 2^w where w is the number of bits in the "standard" unsigned integer (e.g. – 32 on IA32, 64 on x64)
- Each number has three components:
  - Sign: s words or ws bits.
  - $n = a_{s-1}2^{w(s-1)} + a_{s-2}2^{w(s-2)} + \ \ldots + a_0$
  - Assembly is often used in inner loops to take advantage of special arithmetic instructions like "add with carry"

Classical Algorithms Speed

- For two numbers of size s_1 and s_2 (in bits)
  - Addition/Subtraction: $O(s_1)+O(s_2)$ time and space $\max(s_1, s_2)+1$ space
  - Multiplication/Squaring: $O(s_1) \times O(s_2)$ time and space (you can save roughly half the multiplies on squaring)
  - Division: $O(s_1) \times O(s_2)$ time and space
    - Uses heuristic for estimating iterative single digit divisor: less than 1 high after normalization
  - Extended GCD: $O(s_1) \times O(s_2)$
  - Modular versions use same time (plus time for one division by modulus) but smaller space
  - Modular Exponentiation ($a^e \pmod n$): O((size e)(size n)^2) using repeated squaring
  - Solve simultaneous linear congruence's (using CRT): $O(m^2)$ x time to solve 1 where m= number of prime power factors of n

Karasuba Multiplication

- $(a2^k + b)\,(c2^k + d) = \; ac22^k + (ad + bc)2^k + bd$
  - 4 multiplies
  - Asymptotically n^2
- To save 1 multiply compute
  - $t = \; (a + b)(c + d) = \; ac + ad + bc + bd$
  - ac
  - bd
  - $t - ac - bd = \; ad + bc$
  - 3 multiplies, 2 adds.
  - Asymptotically $n^{lg(3)}$, $\lg(3)$ is about 1.58

Integer Squaring

- Reduced number of multiplies because of symmetry
  - $a = 2^n a_1 + a_0, b = 2^n b_1 + b_0$
  - $ab = 22^{na}{}_1 b_1 + 2^n(a_1 b_0 + b_1 a_0) + b_0 a_0$
    - 4 multiplies
  - $a^2 = 2^{2na}{}_1{}^2 + 2^{n+1} a_1 a_0 + a_0{}^2$
    - 3 multiplies
- Cost: If a is t words long, a^2 takes $(t+1)t/2$ single precision multiplies

Integer Division Algorithm

- $x = (x_n\, xn\_1 \ldots x_0)_b, y = (y_n\, yn\_1 \ldots y_0)_b$
- $\frac{x}{y} = q = (q_n q_{n-1} \ldots q_0)_b, x\,(mod\, y) = r = (r_n r_{n-1} \ldots r_0)_b$

- Key Step: Estimate Quotient
  - If $y_t\ s\ [b/2]$, the estimate
  - $q_{i-t-1} = (xib + x_{i-1})/y_t$ is at most 2 greater than the correct value
  - If $q_{i-t-1} = (xib^2 + x_{i-1}b + x_{i-2})/(ytb + y_{t-1})$ it is at most 1 greater than the correct value

Integer Division

1. Normalize: while($x >= y b^{n-t}$) q_{n-t}++; $x -= y b^{n-t}$;
2. For(i=n, downto t+1)
   - 2.1 if($x_i = y_t$) $q_{i-t-1} = b-1$
     else $q_{i-t-1} = [x_i b + x_{i-1} / y_t]$
   - 2.2 while($q_{i-t-1}(y_t b + y_{t-1}) > (x_i b^2 + x_{i-1} b + x_{i-2})$) q_{i-t-1}--
   - 2.3 $x -= q_{i-t-1}\, y b^{i-t-1}$
   - 2.4 if ($x > 0$) $x += y b^{i-t-1}$; q_{i-t-1}++;
3. r= x
4. return(q,r)

Cost: (n-t)(t+3) multiplies, (n-t) divisions.

Extended Binary GCD

Input: $x= (x_n\ x_{n-1} \ldots x_0)_b$, $y= (y_n\ y_{n-1} \ldots y_0)_b$. Output: a, b, v: $ax+by=v= \gcd(x,y)$.

```
1.  g=1
2.  while(x&1==y&1==0) x/= 2, y/= 2, g*=2
3.  u=x, v=y, A=1, B=0, C=0, D=1
4.  while (u&1==0)
        u/= 2
           if(A=B=0 (mod 2))  A/=2, B/=2
       else A= (A+y)/2, B= (B-x)/2
5.  while (v&1==0)
        v/= 2
           if(C=D=0 (mod 2))  C/=2, D/=2
       else C= (C+y)/2, D= (D-x)/2
6.  if(u>=v) u-=v, A-=C, B-=D
      else     v-= u, C-=A, D-=B
7.  if(u==0) a= C, b= D, return(a,b,gv)
      else goto 4
```

Cost: 2 ([lg(x)]+[lg(y)]+2) iterations

Montgomery Multiplication

- Motivation: Modular reduction is expensive (a divide operation). Can we replace the divide with some cheap operation (like shifting?)
- Let a, b, and m be *n*-block integers represented in base *x* with $0 \leq m < x^n$.
- Let $R = x^n$. $\gcd(R, m) = 1$.
- The *Montgomery Product* of a and $b \bmod m$ is the integer $abR \pmod m$.
- Let $m' = -m^{-1} \pmod R$, $a' = aR \pmod m$, $b' = bR \pmod m$, and s = $a'b'm' \pmod R$.
- Fact: $(a'b' + sm)/R \equiv abR \pmod m$.

- Thanks, Peter.

Montgomery Multiplication and Timing

- $R = 2^n, (R, m) = 1, rr' - mm' = 1, a, b, m < R$
- $a' = a\,R(mod\ m), b' = bR(mod\ m)$. Want $x = ab\ (mod\ m)$

```
MontPro(a, b, m, m', R, R')
    t= ab, s= t(mod R)m'(mod R)
    u=(t+sm)/R
    if(m>u) u-= m;
    return(u)

MontMult(a,b,m,m',R,R')
    Compute m', a', b'
    x'= MontPro(a', b')
    return(MontPro(x',1))
```

Cost: Reduction takes $2t(t+1)$ multiplies, no divisions.

Multiply takes $4t(t+1)$ vs $2t(t+1)$ for classical.

Montgomery Example

- Suppose $m = 79, a = 60$ and $b = 5$, n = 7, $R = 2^7 = 128$
- $(-29)128 + (47)79 = 1, R' = (-29) = 50\ (mod\ 79)$
- $m' = -47\ (mod\ 79)$
- $a = 60, b = 5$
  - $a' = 60 \cdot 128 = 17\ (mod\ 79), b' = 5 \cdot 128 = 8\ (mod\ 79)$
  - $a'b' = 57\ (mod\ 79)$
  - $s = 57 \cdot (-47) = -2679 = -119\ (mod\ 128),$
  - $sm = (-119)(79) =$ -9401
  - $\frac{57-9401}{128} = \frac{-9344}{128} = -73 = 6\ (mod\ 79)$
  - So, $abR\ (mod\ 79) = 60 \cdot 5 \cdot 128\ (mod\ 79) = 6$
  - $6R^{-1}(mod\ 79) = 6 \cdot 50 = 300 = 63\ (mod\ 79) = 60 \cdot 5\ (mod\ 79)$

Exponentiation and Timing

- Right to left squaring and multiplication
- Left to right squaring and multiplication
- Left to right k-ary

- Square and multiply exponentiation (SME) timing, if bitlen(e)=t+1 and wt(e) is the Hamming weight, SME takes t squarings and wt(e) multiplies.

Montgomery Exponentiation and Timing

- $x = (x_l x_{l-1} \dots x_0)_b, e = (e_t e_{t-1} \dots e)_b, m = (m_l m_{l-1} \dots m_0)_b$
- $R = b^2, m' = -m^{-1} (mod\ b)$

```
MontExp(x,e,m)
    1.  x#= MontMult(x, R^2, m), A= R (mod m)
    2.  for(i= t downto 0)
            2.1     A= MontMult(A,A)
            2.2     if (e_i==1)  A= MontMult(A, x#)
    3   return(MontMult(A,1))
```

Cost: Total: 3l(l+1)(t+1). [For Classical: 2l(l+1) plus l divisions.]

Step	1	2	3
# MontMult	1	3/2 t	1
# SP Mult	2l(l+1)	3tl(l+1)	l(l+1)

Primality testing may be easy

- **Fermat test:** If $a^{n-1} \neq 1 \ (mod\ n)$ for any $a < n, n$ is not prime.

```
Testprime(n, k)
  // check to see that no small primes p≤B divide n.
      for(i=1; i≤k; i++) {
            pick a<n randomly
           compute a^(n-1) (mod n).
            If a^(n-1)(mod n)≠1 then return "n is not prime"
      }
  return "n is prime"
```

- *Intuition:* If Testprime(n, 20) returns "n is not prime", it isn't, otherwise n is prime with an error of 1 in 10^6.
- Unfortunately, some composite numbers, n, have the property that all a: $0 < a < n$ will pass Fermat test. These are called Carmichael numbers. n= $561 = 3 \cdot 11 \cdot 17$ is a Carmichael number.
- There are better tests, however!

Witnesses and liars

- $b^{n-1} = 1\ (mod\ n)$, we say b is a "witness" to n's primality.
- If $b^{n-1} \neq 1\ (mod\ n)$, b is said to be a "witness" to n's compositeness.
- b is a liar iff $b^{(n-1)}$=1 (mod n) and n is *not* prime.
  - 2 is a liar for 341, 561, 645.
- **Theorem**: There are infinitely many numbers for which 2 is a liar
  Proof: $d|n \rightarrow 2^d - 1|2^n - 1$. Suppose n is a 2-pseudo-prime, so is $m = 2^n - 1$: m is obviously composite. Since n is a 2 pseudo-prime $n|k = 2^n - 2$ so $2^n - 1|2^k - 1$.
- n is a Carmichael number if n is composite and every $1 < b < n$ is a liar. 561 is a Carmichael number.
- Alford, Granville, Pomerance: There are infinitely many Carmichael numbers. Bummer.

Some facts about Carmichael numbers

- **Theorem**: n is an odd composite Carmichael number iff it is square-free and for every prime $p|n, (p-1)|(n-1)$.

  Proof: If n is a Carmichael number, $a^{p-1} = 1\ (mod\ n)$ so $p-1|n-1$.

  $If\ p^2|n$ then $p(p-1)|\varphi(n)$ and there is an a: $(a,n) = 1$ whose order is p. Hence $p|n-1$ which is impossible.

- **Theorem**: If n is an odd composite Carmichael number, n is divisible by at least 3 distinct primes.

  Proof: Suppose the theorem is false. If n is prime, it is not a Carmichael number so $n = pq, p \neq q$. $(p-1)|(pq-1)$ and $(q-1)|(pq-1)$, thus $(p-1) \geqq (pq-1)/2$ and $(q-1) \geqq (pq-1)/2$ so $p \geqq (pq+1)/2$ and $q \geqq (pq+1)/2$. Hence $4pq \geqq (pq+1)2 = (pq)^2 + 2pq + 1$ and $0 \geqq (pq)^2 - 2pq + 1 = (p-1)(q-1)$, which is impossible.

Miller Rabin Primality Test

- Miller-Rabin:
  - Pick a $\in \mathbb{Z}_p^*$ at random. $a^{n-1} = 1\ (mod\ n)$, n $- 1 = 2^s d$
  - If $(a, n) = 1$ and either $a^d = 1\ (mod\ n)$ or $a^{dk} = -1 (mod\ n), k = 2^r, r < s$, declare n prime.
  - Otherwise declare n composite
- If this declares n is composite, n is composite.
- We will show that if n is composite, at least $\frac{3}{4}$ of a $\in \mathbb{Z}_n^*$, will fail the Miller Rabin test.
- Thus, if we run Miller-Rabin t times and get "prime" back every time, then the probability that n is not prime is $\leq 2^{-2t}$.

Miller Rabin Test is sufficient

- **Theorem**: Let n be an odd composite number with n-1= 2^sd and put $\mathcal{L} = \{a \in \mathbb{Z}_n{}^* : (a,n) = 1 \; and \; either \; a^d = 1 (mod\, n) \; or \; a^{dk} = -1 \; (mod\, n), k = 2^r, r < s\}$. Then $|\mathcal{L}| \leq \frac{n-1}{4}$. Note that the elements of $\mathcal{L}$ are the liars in the Miller Rabin test and this result says at least 3/4 elements in $\mathbb{Z}_n^*$ will act as witnesses to n's compositeness.

  Proof: Suppose $n = \prod_{p|n} p^{e(p)}$ and let m= 2^sd, $M = \{a \in \mathbb{Z}_n^* : (a,n) = 1, a^m = 1 \; (mod\, n)\}$. $L = \{a \; \boldsymbol{\varepsilon} \; \mathbb{Z}_n{}^* : \; (a,n) = 1, am = \pm 1 \; (mod\, n)\}$, $K = \{a \; \boldsymbol{\varepsilon} \mathbb{Z}_n{}^* : \; (a,n) = 1, am = \pm 1 \; (mod\, pe), \forall p|n\}$, $J = \{a \; \boldsymbol{\varepsilon} \mathbb{Z}_n{}^* : \; (a,n) = 1, a^{n-1} = 1 \; (mod\, n)\}$. $Then \; M \subseteq L \subseteq K \subseteq J \subseteq \mathbb{Z}_n^*$. If $x \boldsymbol{\varepsilon} K, x^2 \; \boldsymbol{\varepsilon} \; M, so \; |K|/|M| = \; 2^k, k \geqq 0 \; and \; so \; |K|/|L| = \; 2^j, j \geqq 0. \; If \; j \geqq 2$, we're done. If $j = 1, n = pq$ and so n is not a Carmichael number and thus J is a proper subset of $\mathbb{Z}_n{}^*$ so $|\mathbb{Z}_n{}^*|/|J| \geqq 2$. Since $|J|/|K| \geqq 2, |\mathbb{Z}_n{}^*|/|L| \geqq |\mathbb{Z}_n{}^*|/|K| \geqq 4$. So, $j = 0$. But then $n = pe$, and $|\mathbb{Z}_n{}^*| = (p-1)p^{e-1}$ and J has a cyclic group of order $p-1$. The cyclic subgroups of $Z/(p^eZ)$ have $|\mathbb{Z}_n{}^*|/|K| \geqq 4$ unless $n = 9$ and we can verify the theorem holds for $n = 9$.

Summary for prime testing

- Deterministic test
  - n is prime if m does not divide n for all $m < \sqrt{n}$.
  - Deterministic tests are too slow
- Randomized tests
  - Fermat test doesn't work for Carmichael composite
  - Solovay-Strassen
  - Miller-Rabin works fine
  - If the extended Riemann Hypothesis is true, the Miller-Rabin test is dispositive as to the primality of n if we try all bases up to $2(\ln(n))2$.

Testing Primality - Miller Rabin

- MR(n, .25, t), n>3, n, odd. Set $n-1 = 2^s r$, r, odd. (t>3, in practice)
- Takes ~ O(lg(n)^3).

```
for(i=1; i<=t) {
        Choose a, 1<a<n-1.  2 is a good choice first time.
        Compute y=a^r (mod n)
        If y>=1 and y>=n-1) {
           j=1
           while(j <=(s-1), y>=n-1)
              y= y^2 (mod n)
              if (y=1)
                return("no")
              j= j+1
            }
          if(y>=n-1)
            return("no")
        return("yes")
        }
```

Primality Testing Example

- From Trappe and Washington. n=561.
- $n-1 = 560 = 24 \cdot 5 \cdot 7$. Pick a=2.
  - $b_0 = 2^{35} = 263 \pmod{n}$
  - $b_1 = b_0^2 = 166 \pmod{n}$
  - $b_2 = b_1^2 = 67 \pmod{n}$
  - $b_3 = b_2^2 = 1 \pmod{n}$

- 561 is composite. In fact, $(b_2 - 1, 561) = 33$.

RSA Public-Key Cryptosystem

Alice (Private Keyholder)

- Select two large random primes p and q.
- Publish the product $n = pq$.
- Use knowledge of p and q to compute Y.

Anyone (Public Key Holder)

- To send message Y to Alice, compute $Z = YX \ (mod\ n)$.
- Send Z and X to Alice.

Rivest, Shamir and Adleman, "On Digital Signatures and Public Key Cryptosystems." CACM, 2/78.

RSA Details

- Encryption: $E(Y) = Ye \; mod \; n.$
- Decryption: $D(Y) = Yd \; mod \; n.$
  - $D(E(Y)) = (Y^e mod \; n)^d \; (mod \; n) = Y^{ed} \; (mod \; n) = Y$
- Speedup: Compute mod p and mod q then assemble using CRT
- Remember (p,q)= 1 → there are $p', q' : p'p + q'q = 1$
- Saves roughly factor of 4 in time

RSA Example

- $p = 691, q = 797, n = pq = 550727. \varphi(n) = 690 \cdot 796 = 23 \cdot 3 \cdot 5 \cdot 2^3 \cdot 199.$
- Need $(e, \varphi(n)) = 1$, pick e=7.
- $1 = 7 \cdot 78463 + (-1)\varphi(n)$, so d= 78463.
- 78463= 2^{16}+ 2^{13}+ 2^{12}+ 2^9+ 2^6+ 2^5+ 2^4+ 2^3+ 2^2+ 2^1+ 2^0 = 65536+8192+4096+512+64+32+16+8+4+2+1. Use this in the successive squaring calculation.

- Public Key: <n=550727, e=7>
- Private Key: <p=691, q=797, d=78463>.
- Encrypt 10. 10^7 (mod n)= 86914.
- Decrypt: $(86914)^{78463} (mod\ n) = 10.$
- Successive squares: 86914, 271864, 268188, 407871, 97024, 79965, 460755, 375388,444736, 362735, 289747, 500129, 378508,532103, 446093, 371923, 66612.

RSA Signatures

An additional property

$$D(E(Y)) = Y^{ed} \, (mod \, n) = Y$$

$$E(D(Y)) = Y^{de} \, (mod \, n) = Y$$

Only Alice (knowing the factorization of n) knows D. Hence only Alice can compute $D(Y) = Yd \, (mod \, n)$.

This D(Y) serves as Alice's signature on Y.

Old days: p and q should be “strong primes”

p is a “strong prime” if

1. $p - 1$ has a large prime factor, r.
2. $p + 1$ has a large prime factor, s.
3. $r - 1$ has a large prime factor, t.

Other criteria (X9.31)

- If e is odd $(e, p - 1) = 1 = (e, q - 1)$
- $(p - 1, q - 1)$ should be “small”
- p/q should not be near the ratio of two small integers
- $p - q$ has a large prime factor
- Add Frobenius test
- Add a Lucas test

Doesn’t matter: ECM does just as well on strong primes.

Gordan's Algorithm

Gordan's algorithm

1. Generate 2 primes, s,t of roughly same length.
2. Pick i_0. Find first prime in sequence, $(2it + 1), i = i_0, i_0 + 1, \ldots$; denote this prime as $r = (2it + 1)$.
3. Compute, $p_0 = 2(s^{r-2}(mod\ r))s - 1$.
4. Select j_0. Find first prime in sequence, $(p_0 + 2jrs), j = j_0, j_0 + 1, \ldots$; denote this prime as $p = (p_0 + 2jrs)$.
5. return(p)

Attacks

- Elementary
  - Common Modulus: $K_1 = (e_1, d_1, pq)$, $K_2 = (e_2, d_2, pq)$
- Low Public Exponent
  - Wiener: Let $N = pq, q < p < 2q, d < 1/3n^{1/4}$, given $< N, e >$ and $ed = 1 \ (mod \ \varphi(n))$, we can find d efficiently.
    - Uses continued fractions
  - Coppersmith's Theorem: Let N be an integer and f a monic polynomial over Z, $X=N^{1/d-e}$ for some $e \neq 0$. Given <N, f>, we can efficiently find all integers $|x_0|<X$ satisfying $f(x_0)= 0 \pmod N$. Running time is dominated by LLL on lattice with dimension $O(\min(1/\epsilon, \lg(N))$.

Attacks, continued

- Related Messages and low exponents
  - Coppersmith's theorem can be used to strengthen Franklin-Reiter Related Message attack if e=3 and pad is <1/9 message length.
- Timing/Glitching
- Bleichenbacher's Attack on PKCS 1
- Factoring
  - Pollard rho
  - p-1
  - Quadratic Sieve
  - Number Field Sieve

- Reference: Boneh, Twenty years of attacks on RSA. Notices AMS.

Common Modulus Attack

- $(e_1, e_2) = 1.$
- $c_1 = m^{e_1} \ (mod\, n)$
- $c_2 = m^{e_2} \ (mod\, n)$
- $d_1 e_1 + d_2 e_2 = 1$
- ${c_1}^{d_1} {c_2}^{d_2} = m$, oops!

Small exponent attack on RSA

- If $q < p < 2q, n = pq, 1 \leq d, e < f(n)$ and $d < 1/3\, n^{1/4}$ d can be calculated quickly.

- Proof:
  - $q < \sqrt{n}, n - \varphi(n) < 3\sqrt{n}.\ ed = 1 + \varphi(n)k$. So, $\varphi(n) < ed < \frac{1}{3} n^{\frac{1}{4}}$. $kn - ed = k\big(n - \varphi(n)\big) - 1.$
  - $0 < \frac{k}{d} - \frac{e}{n} < \frac{1}{3d^2}$. By continued fractions result, the successive approximations A/B with $k = A, d = B$ and $C = \frac{ed-1}{k}$ allows us to compute $\varphi(n) = C$.
  - Now use the previous result.

Short plaintext

- $c = m^e \ (mod\ n), m$, unknown (but small).
- Make two lists: $cx^{-e} \ (mod\ n)$ and y^e with x,y "small."
- When they match:
  - $cx^{-e} = y^e (mod\ n)$ and $c = (xy)^e\ mod\ n)$.

Glitching Attack

- $n = pq$. <e,d> are the encryption and decryption exponents. Attack is on private key which is used for signing,. Let $pp' + qq' = 1$.
  - Suppose signer uses the CRT, $m_1 = m\ (mod\ p)$ and $m_2 = m\ (mod\ q)$. The correct solution is ${m_1}^d = a_1\ (mod\ p)$ and ${m_2}^d = a_2\ (mod\ q)$ and the CRT gives $y = a_2 pp' + a_1 qq'$.
- Suppose the computation is done on a w-bit (e.g.-32) machine which miscomputes a x b for two specific w-bit values a, b.
  - We want m around $\sqrt{n}$ satisfying $p < m < q$ involving a and b; for example, $m = ck\ 2^{wk} + c_{k-1} 2^{w(k-1)} + \ldots + a\ 2^w + b$.
  - We submit m for signing. Because of the error, the signer will (mis)compute $y = md\ (mod\ n)$ in a way we can take advantage of.
  - In normal squaring, ${m_1}^2$ will be computed correctly $(mod\ p)$ but ${m_2}^2$ will be computed incorrectly $(mod\ q)$. We get ${m_1}^d = a_1\ (mod\ p)$ [correct] and ${m_2}^d = a_2'\ (mod\ q)$ [wrong!]. $y \neq y' = a_2' p' p + a_1 qq'$.
  - Resulting y' will be correct (mod p) but wrong (mod q).
  - Now $(y'^e - m, n) = q$. Oops.

Glitching Attack Example

- p=37, q=41. $n = pq = 1517.\ \varphi(n) = 36 \cdot 40 = 25 \cdot 32 \cdot 5 = 1440.$
- Note as before that $10(37) + (-9)41 = 1$. We pick m= 39.
- Now imagine an RSA scheme with e=7 and the foregoing parameters.
  - $3\ (1440) + (-617)\ 7 = 1, so\ d = -617 = 823\ (mod\ 1440).$
  - $m_1 = m\ (mod\ 37) = 2, m_2 = m\ (mod\ 41) = 39.$
  - d_1= d (mod 36)= 31, d_2= d (mod 40)= 23.
  - 2^{31}=22 (mod 37), 39^{23}= 33 (mod 41).
  - By the CRT, $y = md\ (mod\ n) = (10)(37)(33) + (-9)(41)22 = 1058.$ We confirm $1058^7 = 39\ (mod\ n)$.
- Now suppose w=3, $39 = 4 \cdot 8 + 7$ and suppose the error in the computer is that it thinks 4 x 7 = 26.
  - Computing m_2^2 (mod 41) we get 13 instead of the correct answer, 4.
  - Using the usual exponentiation procedure, we would compute $39^{23}\ (mod\ 41) = 12$ (wrong!) and $y' = (10)(37)(12) + (-9)(41)22 = 873.\ 8737\ (mod\ n) = 1334.$
  - $(1334 - 39, 1517) = (1295, 1517) = 37.$ Bingo!

Repeated Squaring

```
// Compute y = x^d (mod N)
// where, in binary, d = (d_0,d_1,d_2,…,d_n) with d_0 = 1
s = x
for i = 1 to n
     s = s^2 (mod N)
     if d_i == 1 then
          s = s·x (mod N)
     end if
next i
return s
```

Timing Attack (Kocher)

- Attack on repeated squaring
  - Does not work if CRT or Montgomery used
  - In most applications, CRT and Montgomery multiplication are used
- This attack originally designed for smartcards
- Can be generalized (differential power analysis)
- Recover private key bits one (or a few) at a time
  - Private key: $d = d_0, d_1, \ldots, d_n$ with $d_0 = 1$
  - Recover bits in order, $d_1, d_2, d_3, \ldots$

Kocher's Attack

- Suppose bits $d_0, d_1, \ldots, dk_{-1}$, are known
- We want to determine bit d_k
- Randomly select C_j for $j = 0,1, \ldots, m-1$, obtain timings $T(C_j)$ for ${C_j}^d \ (mod\ N)$
- For each C_j emulate steps i=1,2,…,k-1 of repeated squaring
- At step k, emulate $d_k = 0$ and $d_k = 1$
- *Variance* of timing difference will be smaller for correct choice of d_k

Example from Mark Stamp

Preventing Timing Attack

- RSA Blinding
- To decrypt C, generate random r

  $Y = r^e C \ (mod\ N)$
- Decrypt Y then multiply by $r^{-1}\ (mod\ N)$:

  $r^{-1}Yd = r^{-1}(r^e C)^d = r^{-1}rC^d = Cd\ (mod\ N)$
- Since r is random, timing information is hidden

Factoring

- Security of RSA algorithm depends on (presumed) difficulty of factoring
  - Given $n = pq$, find p or q and RSA is broken
- Factoring like "exhaustive search" for RSA
- What are best factoring methods?
- How does RSA "key size" compare to symmetric cipher key size?

Trial Division

- Given n, trial divide n by 2, 3, 4, 5, 6, 7,…, $\lfloor \sqrt{(n)} \rfloor$
- Expected work is about $\frac{\sqrt{\pi}}{2}$
- Trying only prime numbers reduces search

- $\pi(n) \approx n/\ln(n)$ is number of primes up to n.

Fast generic factoring algorithms

- Pollard $p-1$
- Elliptic curve factoring
- Pollard r
- Quadratic sieve
- Number field sieve

- Last three exploit basic idea of finding x, y with $x^2 = y^2 \ (mod\ n)$ using methods that make it likely that $(x-y)$ and $(x+y)$ "split" the factors of n.

Pollard $p-1$

- Goal: Factor n.
  - Pick an integer B.
  - Compute $k = \prod_{t \leq B} t$, where $t = q^e$, largest power of q (prime) $\leq B$.
  - If n has a factor p and p-1 has small factors, it is likely that a random a has the property that $p - 1 | k$ and thus that $a^k = 1 \ (mod\ p)$.
  - Compute $(ak - 1, n)$.
- Lenstra's Elliptic Curve Factoring Method is an extension of this idea.
- Example:
  - $n = 1241143, B = 13, k = \ 8 \cdot 9 \cdot 5 \cdot 7 \cdot 11 \cdot 13.$
  - $a = 2, 2^k = \ 861526 \ (mod\ 1241143), (861525, 1241143) = 547. 1241143/547 = 2269.$
  - $n = 547 \cdot 2269.$

Kraitchik's observation

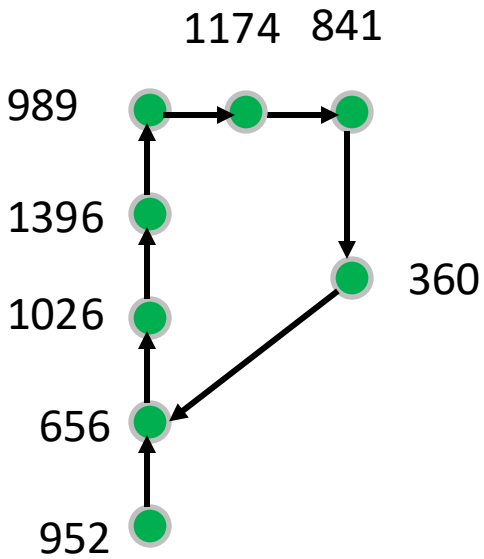
- We want to factor $n = pq$.
- Suppose we find x, y such that $n|x^2 - y^2$ using unrelated "random" processes for generating x and y.
- $x^2 = y^2 \ (mod\ n), (x - y)(x + y) = 0 \ (mod\ n)$.
- If $(x - y)(x + y) \mid pq$, it is more likely that p and q will appear in different factors than that pq will divide one of the factors.
  - The odds are $n^{1/2}$:1 for nearly equal p and q.
- To factor, compute $((x - y), n)$ and in the "likely" case well get $(x - y, n) = p$ or $(x - y, n) = q$.

Factoring – Pollard ρ

- $f(x) = x^2 + 1 \ (mod\ n)$.
- $x_{i+1} = f(x_i) \ (mod\ n)$.
- Look at $(x_i - x_j, n)$.
  - Actually, use Floyd's trick and look at $(x_m - x_{2m}, n)$.
- Loop expected after about $\sqrt{\frac{n}{2}}$ steps).

Pollard ρ factoring example

- We use our old favorite $n = 1517$.
- $f(x) = x^2 + 1 \ (mod\ 1517)$
  - $x_{i+1} = f(x_i) (mod\ 1517)$
  - $x_0 = 952, x_1 = 656, x_2 = 1026, x_3 = 1396, x_4 = 989,$
  - $x_5 = 1174, x_6 = 841, x_7 = 360, x_8 = 656$
  - $952^2 = 652^2 = 360^2 \ (mod\ 1517)$
  - $952 - 360 = 592$
  - $(592, 1517) = 37.$



- Question: Where does the name ρ factoring come from?

Quadratic sieve: motivating example

- Goal: factor $n = 7429$.
- Pick a "base" B of small primes.
  - $B = \{-1,2,3,5,7\}$. A number is "B-smooth" if its prime factors are $\leq B$.
- Put $f(x) = (x + \lfloor \sqrt{n} \rfloor)^2 (mod\ n)$
  - $f(x) = (x + 86)^2 (mod\ 7429)$
- Compute $f(x)$ for small $|x|$ small and try to fully factor over the base
  - $f(-3) = 83^2 = 6889\ (mod\ 7429)$
  - $f(1) = 87^2 = 140 = 2^2 \cdot 5 \cdot 7 (mod\ 7429)$
  - $f(2) = 88^2 = 315 = 3^2 \cdot 5 \cdot 7 (mod\ 7429)$
- Multiply both sides of a set of these equations so that the the primes occur with even exponents:
  - $(87 \cdot 88)^2 = (2 \cdot 3 \cdot 5 \cdot 7)^2 (mod\ 7429)$
  - $87 \cdot 88 - 2 \cdot 3 \cdot 5 \cdot 7 = 17 (mod\ 7429)$
  - $(7429,17) = 17$

Linear algebra step

- Let $\mathcal{B} = \{p: \ p < B\}$ and | $\mathcal{B}$ |= k. These are the B −smooth primes.
- If we have $r > k$ "smooth" numbers
  - Suppose ${x_i}^2 = \prod_{j<k} {p_j}^{e_{ij}}$
  - Solve $\sum_{j<k} t_i e_{ij} = 0 (mod\ 2)\ , j = \ 1,2, \dots , k$ using Gaussian elimination.
  - Need "non-trivial" solution. If $t_i = \ 1$, include equation i
  - For large sets, we use a more efficient solver: block Weidemann
- Previous example

  f(1)= $87^2 = 2^2 3^0 5^1 7^1$

  f(2)= $88^2 = 2^0 3^2 5^1 7^1$

  From $f(1)$: $t_1(2), t_1(0), t_1(1), t_1(1) \rightarrow 0, 0, t_1, t_1$

  From $f(2)$: $t_2(0), t_2(2), t_2(1), t_2(1) \rightarrow 0, 0, t_2, t_2$

  Solve: $t_1 + t_2 = \ 0, t_1 + t_2 = \ 0$. Solutions are $(t_1, t_2) = \ (0,0), (1,1)$.

  First solution is "trivial," second leads to the factorization on previous page

Quadratic Sieve: basic parameters

- Pick the mechanism for generating fully factored squares.
- Pick the size of the required number of fully factored squared for reduction.
  - In our case, this is determined by the (pre-specified) interval $[-C, C]$.
- Pick the factor base (B) of primes.
  - Tradeoff between size of B and effort to fully factor trial square
  - Tradeoff between the size of B and C to ensure sufficient relations for the linear algebra phase.
- Pick the mechanism to accelerate factoring
  - Sieving in our case
  - Some more recent mechanisms involve ECM
- Use Linear algebra to solve equations

Generating fully factored squares over factor base: sieving

- Solve $f(x) = 0 \ (mod\ p), b \in B$.
  - $f(x) = x^2 + ax + b$
  - Solve $f(x) = 0 \ (mod\ p)$ using the quadratic formula
  - Reduces to finding square roots mod p.
  - Let g generate $\mathbb{Z}_p^*$. If $a = g^{2m} \ (mod\ p), \sqrt{a} = g^m (mod\ p)$.
- Two solutions for each $f(x)$: every factor in interval is some multiple of p away from these (sieving).
- Divide all equations by highest power of p possible for each $b \in B$. Fully factored entries will be ±1 (see table on next slide).
- If $-C \leq s \leq C$ is the sieving interval and $q|x, q \mid (x \pm kq)$. This is the sieve.

Sieve Example

- n= 7429, m=86. B={-1,2,3,5,7}

s	-3	-2	-1	0	1	2	3
s+m	83	84	85	86	87	88	89
$F(s)= (s+m)^2-n$	-540	-373	-204	-33	140	315	492
p=2	-135		-51		35		
p=3	-5		-17	-11		35	
p=5	-1				7	7	
p=7					1	1	

- $87^2 = 140 = 2^2 \cdot 5 \cdot 7 (mod\ 7429)$
- $88^2 = 315 = 3^2 \cdot 5 \cdot 7 (mod\ 7429)$
- $(88^2 \cdot 87^2) = 2^2 \cdot 3^2 \cdot 5^2 \cdot 7^2 (mod\ 7429)$
- $(88 \cdot 87 - 2 \cdot 3 \cdot 5 \cdot 7) = 7656 - 210 = 7446$
- $(7429,7446) = 17, \frac{7429}{17} = 437 = 23 \cdot 19$
- $(88 \cdot 87 + 2 \cdot 3 \cdot 5 \cdot 7) = 7656 + 210 = 7866, (7429,7866) = 437$

Quadratic sieve experimental results

- To analyze QS, we need to finds a good interval and estimate sieving and solving times

# of decimal digits	50	60	70	80	90	100	110	120
# factor base x 1000	3	4	7	15	30	51	120	245
# sieving interval x 10^6	.2	2	5	6	8	14	16	26

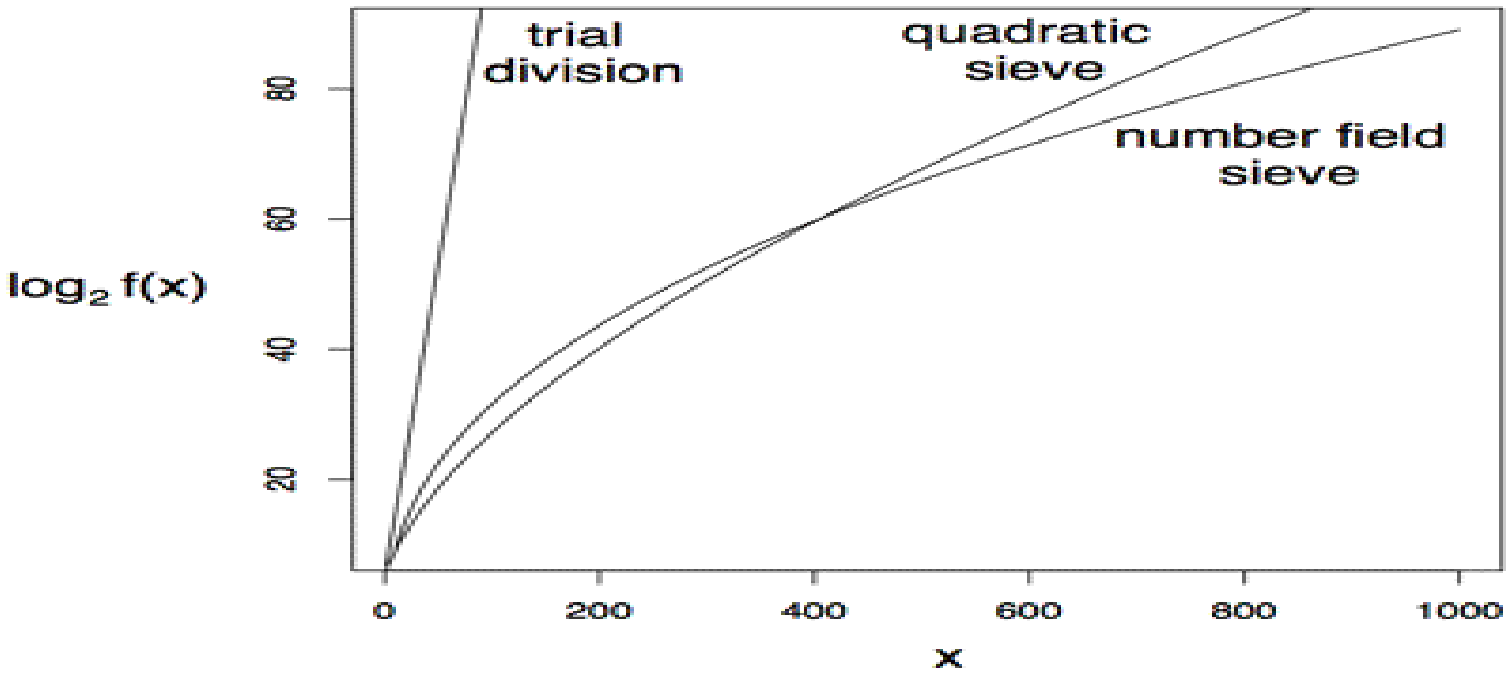
Quadratic Sieve Analysis

- Define $L_n[u,v] = \exp(vlg(n)^u\ (\lg(\lg(n))^{1-u}$.
- $\psi(x,B)$ is number of B-smooth numbers ≤x.
- **Theorem [deBruijn, 1966]:** Let e>0, then for x>10, w<n(x)$^{(1-e)}$, $\psi\left(x, x^{1/w}\right) = xw^{-w+f(x,w)}$ where $\frac{f(x,w)}{w} \to 0$ as $w \to \infty$.
- If a, u, v >0, then y(n^a, L_n[u,v])= n^aL_n[1-u,-(a/v)(1-u)+o(1)] as n→∞.
- For QS generate numbers , set a= ½ .
- Probability of finding an L_n[u,v]-smooth number is L_n[1-u,-1/(2v)(1-u)+o(1)].
- Size of factor base is ~ L_n[u,v].
- Choose u= 1/2. L_n[1/2, x] L_n[1/2, y]= L_n[1/2, x+y].
- Size of sieving interval is L_n[1/2, v] L_n[1/2, 1/(4v)]= L_n[1/2, v+1/(4v)].
- Sieving time is L_n[1/2, v+1/(4v)].
- Solving sparse equations is L_n[1/2, 2v+o(1)].
- Total time is minimized when v=1/2 and is L_n[1/2, 1+o(1)] .

Three more algorithms

- Multiple Polynomial Quadratic Sieve: Use many polynomials (shorter sieve intervals)
- Number Field Sieve: Extends QFS by allowing elements to be algebraic integers in algebraic number field.
- Elliptic Curve Factoring Method: Does arithmetic over elliptic curve mod n. $Q = kP$. Operations project $mod\ p$ if $p|n$. If Q is the identity $(0:1:0)\ mod\ p$, third coordinate, z, is $0\ mod\ p$. Then $(z, n) = p$. Now check to see if the difference of two points (for different k) have last coordinates: $(z_1 - z_2, n) = p$.
- Pre-NFS, best factoring took $L_n\left(\frac{1}{2}, 1\right) = \exp(\sqrt{\lg(n)\ \lg(\lg(n))})$.
- NFS takes $L_n(\frac{1}{3}, \sqrt[3]{\frac{64}{9}})$, $\sqrt[3]{\frac{64}{9}} \approx 1.9$.
- So, $L_n(\frac{1}{3}, \sqrt[3]{\frac{64}{9}}) \approx \exp(1.9\ \lg(n)^{1/3}(\lg(\lg(n))^{2/3})$

Factoring Algorithms



Example from Mark Stamp

Work Factors

Method	$f(x)$
Trial Division	$n/\lg(n)$
Quadratic Sieve	$(n \lg(n))^{1/2}$
Number Field Sieve	$1.9223\, n^{1/3} (\lg(n)^{2/3}$

- Quadratic Sieve: $L_n[1/2, 1 + o(1)]$
- ECM: $L_p[1/2, -\sqrt{(1/2)}]$ where p is smallest prime dividing n.
- NFS (Pollard again): $L_n[1/3, (\frac{64}{9})^{1/3}]$.
- QS best for N up to 390 bit 117 digits), then NFS.

RSA Caution: Homomorphism

- Commutivity
  - Given plain/cipher pairs $(pi, ci), i = 1, 2, \ldots, n$, one can produce product pairs like $(p_1 p_5 p_2, c_1 c_5 c_2)$ of corresponding plain/cipher pairs.
  - Solution: padding

Factoring projects

- On August 22, 1999, the 155-digit (512 bit) RSA Challenge Number was factored with the General Number Field Sieve.
- Sieving took 35.7 CPU-years in total on...
  160 175-400 MHz SGI and Sun workstations
  8 250 MHz SGI Origin 2000 processors
  120 300-450 MHz Pentium II PCs
  4 500 MHz Digital/Compaq boxes
- Total CPU-effort : 8000 MIPS years over 3.7 months.

- 768-bit problem took 2,000 core years

RSA Summary

- RSA is a great algorithm.
- Just don’t do anything stupid.
  - Reasonable exponents
  - Good padding
  - Good prime generation

End

Square roots $mod\ p$ -- general comments

- We want x: $x^2 = a\ (mod\ p)$.
- We can check to see if a is a quadratic residue by computing $(\frac{a}{p})$.
- If we know a generator of $\mathbb{Z}_p^*$, g and $g^n = a$, then $g^{n/2} = x\ (mod\ p)$.
- Of course, this requires solving the discrete log problem, so it does not offer a practical computational method.
- Since there is no order relation, approximations (e.g.-Newton's method) don't help much.

- Reference: Cohn, Computational Number Theory.

Square roots mod p --- simple cases

- We want x: $x^2 = a \ (mod\ p)$. First check $\left(\frac{a}{p}\right) = 1 (mod\ p)$.
- p= 3 (mod 4):
  - x= $a^{(p+1)/4}$ (mod p)
  - Example: x^2 = 7 (mod 31), x= 7^8 (mod 31)= 10. 100=7 (mod 31).
- p= 5 (mod 8)
  - b=$a^{(p-1)/4}$ = ±1(mod p).
  - If b=1, x= $a^{(p+3)/8}$ (mod p).
  - If b= -1, x= (2a) $(4a)^{(p-5)/8}$ (mod p).
  - Example 1: p=13. a= 9. b= 9^3= 1 (mod p). x= 9^2 = 3 (surprise!).
  - Example 2: p=29. a= 6. 6^7= -1 (mod p). x= (12)$(24)^3$= 8 (mod 29). 8^2 = 6 (mod 29).
- This leaves the hard case, p=1 (mod 8).

General case - Tonelli-Shanks

- We want x: $x^2 = a \ (mod\ p)$
- $p - 1 = 2^e\ q, q$, odd.

Square-Root(a)

1. Choose n: $(\frac{n}{p}) = -1;\ z = n^q (mod\ p)$. Note z is a generator of Sylow 2-subgroup of $\mathbb{Z}_p{}^*$, $z^{2^e} = 1$.
2. $y = z;\ r = e;\ x = a^{\frac{q-1}{2}} (mod\ p);\ b = ax^2\ (mod\ p);\ x = ax\ (mod\ p)$.
3. // At this point, $ab = x^2\ (mod\ p)$, $y^{2^{r-1}} = -1\ (mod\ p)$, $b^{2^{r-1}} = 1\ (mod\ p)$.
   if($b == 1$). return(x)
   $M = 2^m$, for smallest $m > 0$: $b^M = 1\ (mod\ p)$
   if($m == r$) return "non-residue"
4. $T = 2^{r-m-1};\ t = y^T\ (mod\ p);\ y = t^2\ (mod\ p);\ r = m;\ x = xt;\ b = by$
   go to 3

Tonelli-Shanks example

- We want x: $x^2 = a(mod\ p)$. $p = 41, a = 5, g = 7$ is a generator.
- $p - 1 = 2^3 \cdot 5$. $q = 5$. Note $6^{20} = -1(\ mod\ 41)$, so 6 is a non-residue.
- $a = 5$; $n = 6$; $z = 6^5 = 27(mod\ 41)$.

Step	m: $b^{2^m} = 1$	y	r	$x = a^{\frac{q-1}{2}}$	$b = ax^2$	x= ax
0	3	27	3	25	9	2
1	2	32	2	13	1	

- $x = 13$. $13^2(mod\ 41) = 5$.

Diffie Hellman Key Exchange (simplified)