Factoring

Factoring

- Security of RSA algorithm depends on (presumed) difficulty of factoring
  - Given N = pq, find p or q and RSA is broken
  - Rabin cipher also based on factoring
- Factoring like "exhaustive search" for RSA
- Lots of interest/research in factoring
- What are best factoring methods?
  - How does RSA "key size" compare to symmetric cipher key size?

Factoring Methods

- Trial division
  - Obvious method but not practical
- Dixon's algorithm
  - Less obvious and much faster
- Quadratic sieve
  - Refinement of Dixon's algorithm
  - Best algorithm up to about 110 decimal digits
- Number field sieve
  - Best for numbers greater than 100 digits
  - We only briefly mention this algorithm

Trial Division

- Given N, try to divide N by each of 2,3,5,7,9,11,…,⌊sqrt(N)⌋
- As soon as a factor found, we are done
  - So, expected work is about sqrt(N)/2
- Improvement: try only prime numbers
- Work is then on order of $\pi(N)$
  - Where $\pi(N) \approx N/\ln(N)$ is number of primes up to N

Congruence of Squares

- We want to factor $N = pq$
- Suppose we find x,y such that $N = x^2 - y^2$
- Then $N = (x - y)(x + y)$, have factored N
- More generally, **congruence of squares**...
- Suppose $x^2 = y^2 \pmod N$
- Then $x^2 - y^2 = kN$ for some k
- Which implies $(x - y)(x + y) = kN$

Congruence of Squares

- Suppose $x^2 = y^2 \pmod N$
- Then $(x - y)(x + y) = kN$
- Implies $(x - y)$ or $(x + y)$ is factor of N
  - Or $x - y = k$ and $x + y = N$ (or vice versa)
- With probability at least 1/2, we obtain a factor of N
  - If so, gcd(N, x – y) or gcd(N, x + y) factors N
  - And the gcd is easy to compute

Congruence of Squares

- For example $10^2 = 3^2 \pmod{91}$
- That is, $(10 - 3)(10 + 3) = 91$
  - Factors of 91 are, in fact, 7 and 13
- Also, $34^2 = 8^2 \pmod{91}$
  - Then $26 \cdot 42 = 0 \pmod{91}$ and we have $\gcd(26,91) = 13$ and $\gcd(42,91) = 7$
- In general, gcd is necessary

Congruence of Squares

- Find congruence of squares: $x^2 = y^2 \pmod N$ and we can likely factor N
- How to find congruence of squares?
- Consider, for example,

  $41^2 = 32 \pmod{1649}$ and $43^2 = 200 \pmod{1649}$
- Neither 32 nor 200 is a square
- But $32 \cdot 200 = 6400 = 80^2$
- Therefore, $(41 \cdot 43)^2 = 80^2 \pmod{1649}$

Congruence of Squares

- Can combine non-squares to obtain a square, for example

  $32 = 2^5 \cdot 5^0$ and $200 = 2^3 \cdot 5^2$

- And $32 \cdot 200 = 2^8 \cdot 5^2 = (2^4 \cdot 5^1)^2$
- We obtain a perfect square provided each exponent in product is **even**
- Only concerned with exponents and only need consider even or odd, i.e., mod 2

Congruence of Squares

- Number has an exponent vector
- For example, first element of vector is power of 2 and second power of 5
- Then

$$32 \rightarrow \begin{bmatrix} 5 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \pmod 2 \qquad 200 \rightarrow \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \pmod 2$$

- And $32 \cdot 200 \rightarrow \begin{bmatrix} 8 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \pmod 2$

Congruence of Squares

- Mod 2 exponent vector of product $200 \cdot 32$ is all zero, so perfect square
- Also, this vector is **sum** (mod 2) of vectors for 200 and 32
- Any set of exponent vectors that sum to all-zero, mod 2, gives us a square
- We need to keep vectors small
  - Only allow numbers with "small" prime factors

Congruence of Squares

- Choose bound B and primes less than B
  - This is our **factor base**
  - For technical reasons, include "–1" in factor base
- A number that factors completely over the factor base is **B-smooth**
- Smooth relations factor over factor base
- Restrict our attention to B-smooth relations
  - Good: Exponent vectors are small
  - Bad: Harder to find relations

Example

- Want to factor N = 1829
- Choose bound B = 13
- Choose factor base –1,2,3,5,7,11,13
- Look at values in –N/2 to N/2
- To be systematic, we choose ⌊sqrt(kN)⌋ and ⌈sqrt(kN)⌉ for k = 1,2,3,4
- And test each for B-smoothness

Example

- Compute

$$42^2 = 1764 = -65 = -1 \cdot 5 \cdot 13 \pmod{1829}$$
$$43^2 = 20 = 2^2 \cdot 5 \pmod{1829}$$
$$60^2 = 1771 = -58 = -1 \cdot 2 \cdot 29 \pmod{1829}$$
$$61^2 = 63 = 3^2 \cdot 7 \pmod{1829}$$
$$74^2 = 1818 = -11 = -1 \cdot 11 \pmod{1829}$$
$$75^2 = 138 = 2 \cdot 3 \cdot 23 \pmod{1829}$$
$$85^2 = 1738 = -91 = -1 \cdot 7 \cdot 13 \pmod{1829}$$
$$86^2 = 80 = 2^4 \cdot 5 \pmod{1829}$$

- All are B-smooth except 60^2 and 75^2

Example

- Obtain exponent vectors

$$42^2 = -65 \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad 43^2 = 20 \rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad 61^2 = 63 \rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$74^2 = -11 \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad 85^2 = -91 \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \quad 86^2 = 80 \rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Example

- Find collection of exponent vectors that sum, mod 2, to zero vector
- Vectors corresponding to 42^2, 43^2, 61^2 and 85^2 work

$$\begin{bmatrix}1\\0\\0\\1\\0\\0\\1\end{bmatrix} \oplus \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix} \oplus \begin{bmatrix}0\\0\\0\\0\\1\\0\\0\end{bmatrix} \oplus \begin{bmatrix}1\\0\\0\\0\\1\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\end{bmatrix}$$

Example

- Implies that

  $(42 \cdot 43 \cdot 61 \cdot 85)^2 = (2 \cdot 3 \cdot 5 \cdot 7 \cdot 13)^2 \pmod{1829}$

- Simplifies to $1459^2 = 901^2 \pmod{1829}$
- Since $1459 - 901 = 558$, we find factor of 1829 by $\gcd(558,1829) = 31$
- Easily verified $1829 = 59 \cdot 31$

Example

- A systematic way to find set of vectors that sum to zero vector…
- In this example, want $x_0, x_1, \ldots, x_5$

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \pmod 2$$

- This is a basic linear algebra problem

Linear Algebra

- Suppose n elements in factor base
  - Factor base includes "–1"
  - Then matrix on previous slide has n rows
  - Seek **linearly dependent** set of columns
- Theorem: If matrix has n rows and n + 1 or more columns then a linearly dependent set of columns exists
- Therefore, if we find n + 1 or more smooth relations, we can solve the linear equations

Dixon's Algorithm

1. To factor N: select bound B and factor base with n−1 primes less than B and "−1"
2. Select r, compute $y = r^2 \pmod N$

   Number r can be selected at random
3. If y factors completely over factor base, save mod 2 exponent vector
4. Repeat steps 2 and 3 to obtain n+1 vectors
5. Solve linear system and compute gcd

Dixon's Algorithm

- If factor base is large, easier to find B-smooth relations
  - But linear algebra problem is harder
- Relation finding phase parallelizable
  - Linear algebra part is not
- Next, quadratic sieve
  - An improved version of Dixon's algorithm

Quadratic Sieve

- Quadratic sieve (QS) algorithm
  - Dixon's algorithm "on steroids"
- Finding B-smooth relations beefed up
- As in Dixon's algorithm
  - Choose bound B and factor base of primes less than B
  - Must find lots of B-smooth relations

Quadratic Sieve

- Define quadratic polynomial

  $Q(x) = (\lfloor \text{sqrt}(N) \rfloor + x)^2 - N$

- The is the "quadratic" in QS
- Use Q(x) to find B-smooth values
  - For each $x \in [-M,M]$ compute $y = Q(x)$
  - Mod N, we have $y = z^2$, where $z = \lfloor \text{sqrt}(N) \rfloor + x$
  - Test y for B-smoothness
  - If y is smooth, save mod 2 exponent vector

Quadratic Sieve

- Advantage of QS over Dixon's is that by using Q(x) we can sieve
- What is sieving? Glad you asked…
- First, consider sieve of Eratosthenes
  - Used to sieve for prime numbers
- Then modify it for B-smooth numbers

Sieve of Eratosthenes

- To find prime numbers less than M
- List all numbers 2,3,4,…,M–1
- Cross out all numbers with factor of 2, other than 2
- Cross out all numbers with factor of 3, other than 3, and so on
- Number that "fall thru" sieve are prime

Sieve of Eratosthenes

- To find prime numbers less than 31…

	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~

- Find that primes less than 31 are

  2,3,5,7,11,13,17,19,23 and 29

Sieve of Eratosthenes

- This sieve gives us primes
- But also provides info on non-primes
- For example, 24 marked with "–" and "/" so it is divisible by 2 and 3
- Note: we only find that 24 is divisible by 2, not by 4 or 8

Sieving for Smooth Numbers

- Instead of crossing out, we divide by the prime (including prime itself)

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30

- **All** 1s represent 7-smooth numbers
- **Some** non-1s also 7-smooth
  - Divide out highest powers of primes

Quadratic Sieve

- QS uses similar sieving strategy as on previous slide
  - And some computational refinements
- Suppose p in factor base divides Q(x)
  - Then p divides Q(x + kp) for all $k \neq 0$ (homework)
  - That is, p divides Q of …,x–2p,x–p,x,x+p,x+2p,…
  - No need to test these for divisibility by p
- This observation allows us to sieve

Quadratic Sieve

- One trick to speed up sieving
- If Q(x) divisible by p, then $Q(x) = 0 \pmod{p}$
- Defn of Q implies $(\lfloor \text{sqrt}(N) \rfloor + x)^2 = N \pmod{p}$
- Square roots of N (mod p), say, s_p and $p - s_p$
  - Let $x_0 = s_p - \lfloor \text{sqrt}(N) \rfloor$ and $x_1 = p - s_p - \lfloor \text{sqrt}(N) \rfloor$
  - Then $Q(x_0)$ and $Q(x_1)$ divisible by p
  - Implies $Q(x_0 + kp)$ and $Q(x_1 + kp)$ divisible by p
- Efficient algorithm for these square roots

Quadratic Sieve

- How to sieve for B-smooth relations
- Array: Q(x) for x = –M,–M+1,…, M–1,M
- For first prime p in factor base
  - Generate all $x \in [-M,M]$ for which p divides Q(x) (as described on previous slide)
  - For each, divide by highest power of p
  - For each, store power, mod 2, in vector for x
- Repeat for all primes in factor base
- Numbers reduced to 1 are B-smooth

Quadratic Sieve

- Linear algebra phase same as Dixon's
- Sieving is the dominant work
- Lots of tricks used to speed up sieving
  - For example, "logarithms" to avoid division
- Multiple Polynomial QS (MPQS)
  - Multiple polynomials of form $(ax+b)^2 - N$
  - Can then use smaller interval $[-M,M]$
  - Yields much faster parallel implementations

Sieving Conclusions

- QS/MPQS attack has two phases
- Distributed relation finding phase
  - Could recruit volunteers on Internet
- Linear equation solving phase
  - For big problems, requires a supercomputer
- **Number field sieve** better than QS
  - Requires 2 phases, like QS
  - Number field sieve uses advanced math

Factoring Algorithms

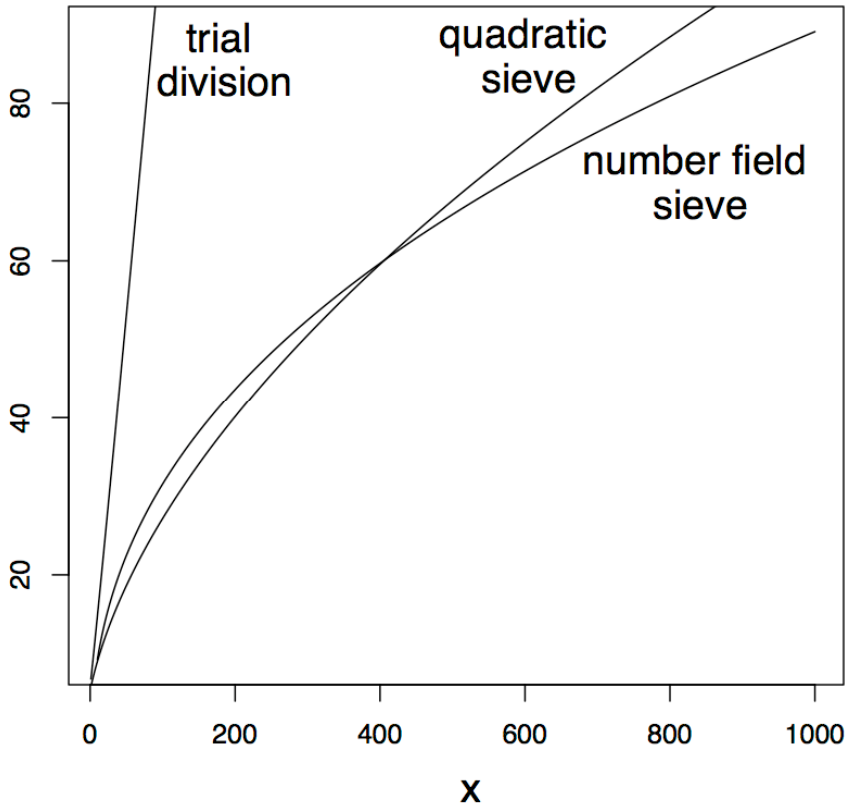
- Work to factor N = 2^x

Factoring Method	$f(x)$	$\log_2 f(x)$
Trial division	$2^x/x$	$x - \log_2 x$
Quadratic sieve	$2^{x^{1/2}(\log_2 x)^{1/2}}$	$x^{1/2}(\log_2 x)^{1/2}$
Number field sieve	$2^{1.9223\, x^{1/3}(\log_2 x)^{2/3}}$	$1.9223\, x^{1/3}(\log_2 x)^{2/3}$

- Last column measures "bits" of work
- Symmetric cipher exhaustive key search: x bit key is x–1 bits of work

Factoring Algorithms

- Comparison of work factors
- QS best to 390 bit N
  - 117 digits
- 390-bit N is as secure as 60-bit key



Factoring Conclusions

- Work for factoring is **subexponential**
  - Better than exponential time but worse than polynomial time
  - Exhaustive key search is exponential
- Factoring is active area of research
  - Expect to see incremental improvement
- Next, discrete log algorithms…