DTTF/NB479: Dszquphsbqiz Day 23

- Announcements:
  1. Term project groups and topics due tomorrow midnight
     - Waiting for posts from most of you.
- Questions?
- This week:
  - Primality testing, factoring
  - Discrete Logs

Factoring

- If you are trying to factor $n=pq$ and know that $p \sim q$, use *Fermat factoring*:
  - Compute $n + 1^2$, $n + 2^2$, $n + 3^2$, until you reach a perfect square, say $r^2 = n + k^2$
  - Then $n = r^2 - k^2 = (r+k)(r-k)$
- Example: factor 2405597
- The moral of the story?
  - Choose p and q such that _____

(p-1) Algorithm

- Useful if $p|n$ and $(p-1)$ has only small factors
- Choose any $a>1$ (like $a=2$) and a bound B
- Compute $b=a^{B!} \pmod n$ (How?)
- Then compute $d=\gcd(b-1, n)$
  - If $1<d<n$, then d is a non-trivial factor

- Matlab example: $n=5183$. We'll use $a=2$, $B=6$.
- Why does it work?

Moral of this story?

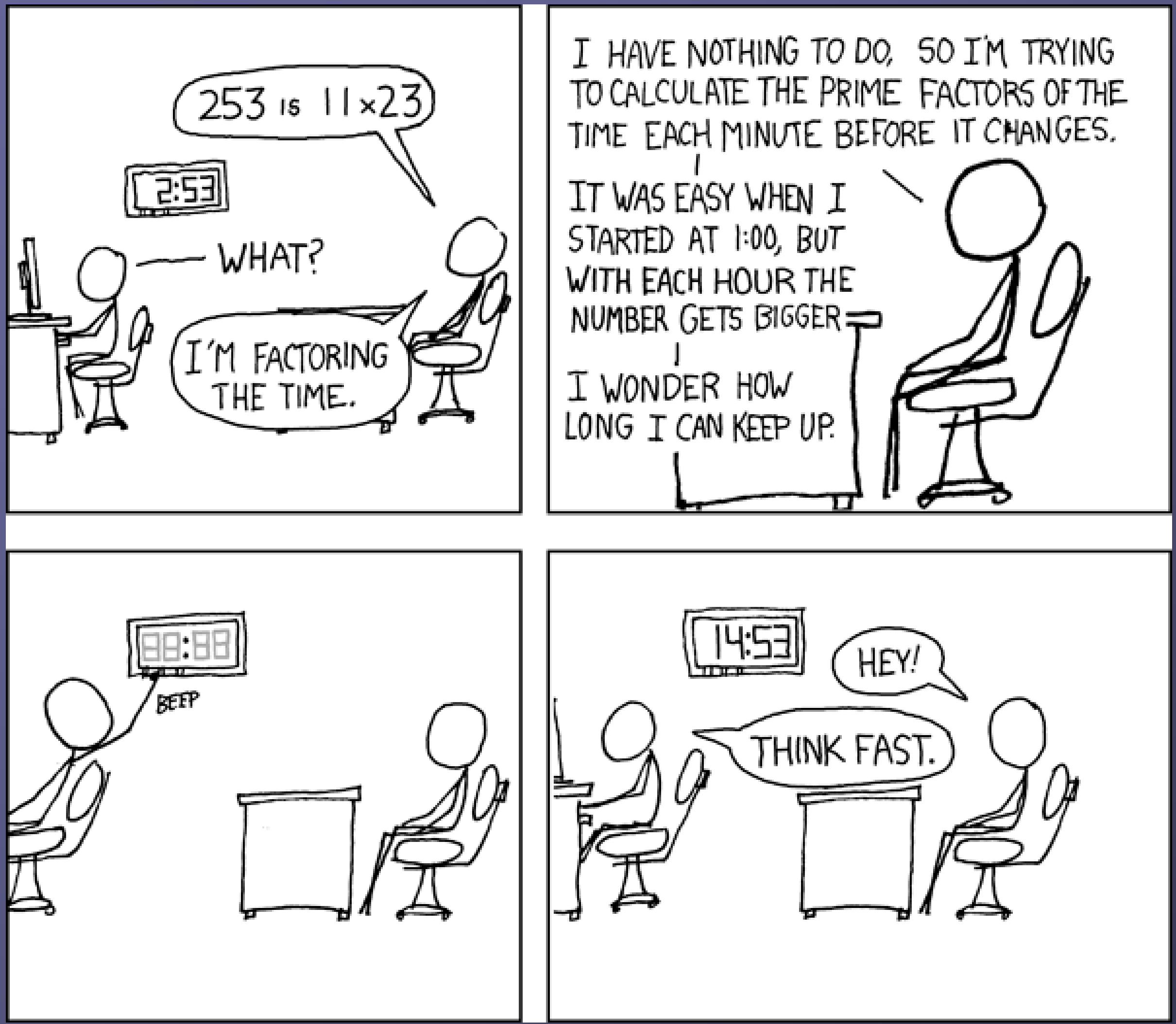
- To get a 100-digit number n=pq resistant to this attack:
  - Make sure (p-1) has at least 1 large prime factor:
  - Pick $\mathbf{p_0 = nextprime(10^{40})}$
  - Choose $k \sim 10^{60}$ such that $\mathbf{p=(kp_0+1)}$ is prime
    - How to test?
  - Repeat for q.

Summary of known implementation mistakes

- Choosing p and q close to each other
- Choosing p and q such that (p-1) or (q-1) has only small prime factors
- Choosing e=3 (smallest e such that gcd(e,(f(n))=1 (problem 6.8.10 and 6.9.14)
- Using a scheme such that ½ the digits of p or q are easy to find (6.2 Theorem 1)
- Choosing e too small (6.2 Theorem 2)
- Choosing d too small ($d < 1/3\, n^{1/4}$; 6.2 Theorem 3; exposes to continued fraction attack)
- Choosing plaintext much shorter than n
  - (But can pad plaintext; see scheme on p. 173)
- One of the factoring Bonus problems suffers from one such mistake

Summary so far: Two of three factoring methods

1. *Fermat factoring*:
   - Compute $n + 1^2$, $n + 2^2$, $n + 3^2$, until you reach a perfect square, say $r^2 = n + k^2$
   - Then $n = r^2 - k^2 = (r+k)(r-k)$
2. (p-1) algorithm:
   - If (p-1) has only small factors, one can factor n:
   - Compute $b=a^{B!}(\mathrm{mod}\ n)$, then $d=\gcd(b-1, n)$ is a factor.
   - How to avoid this?
3. Quadratic sieve (next)



http://xkcd.com/247/

I occasionally do this with mile markers on the highway

Example

Factor $n = 3837523$

- Concepts we will learn also apply to factoring really big numbers. They are the basis of the best current methods
- All you had to do a couple years ago to win $30,000 was factor a 212 digit number.
- This *was* the RSA Challenge: http://www.rsa.com/rsalabs/node.asp?id=2093#RSA704

Quadratic Sieve (1)

Factor n = 3837523

Want x,y: $x^2 \equiv y^2, but\ x \neq \pm y \pmod{n}$ → gcd(x-y, n) is a factor

Step 1: Pick a *factor base*, just a set of small factors.

- In our examples, we'll use those < 20.
- There are eight: 2, 3, 5, 7, 11, 13, 17, 19

Quadratic Sieve (2)

Factor n = 3837523

Want x,y: $x^2 \equiv y^2, \mathit{but}\ x \neq \pm y \pmod{n}$ → gcd(x-y, n) is a factor

Step 2: We want squares that are congruent to products of factors in the factor base.

For example, we note that 8077^2 mod(n) = 2 * 19

Demo Matlab

Quadratic Sieve (2a)

Factor n = 3837523

Want x,y: $x^2 \equiv y^2, \text{ but } x \neq \pm y \pmod{n}$ → gcd(x-y, n) is a factor

Step 2: We want squares that are congruent to products of factors in the factor base.

Our hope: Reasonably small numbers are more likely to be products of factors in the factor base.

$$\text{Want } x^2 = kn + \varepsilon, \text{ so approximate with } x = \left\lfloor \sqrt{kn} + \varepsilon \right\rfloor$$

1. Then $x^2 \approx kn + 2\varepsilon\sqrt{kn} + \varepsilon^2$ which is small as long as k isn't too big
2. Loop over small ε, lots of k.
3. A newer technique, the *number field sieve,* is somewhat faster

Quadratic Sieve (2b)

Factor n = 3837523

Want x,y: $x^2 \equiv y^2, \, but \; x \neq \pm y \pmod{n}$ → gcd(x-y, n) is a factor

Step 2: We want squares that are congruent to products of factors in the factor base.

Our hope: Reasonably small numbers are more likely to be products of factors in the factor base.

$$Want \; x^2 = kn + \varepsilon, so \; approximate \; with \; x = \left\lfloor \sqrt{kn} + \varepsilon \right\rfloor$$

Examples:

$$8077 = \sqrt{17n} + 1; \quad 8077^2 \equiv 38 = 2 \cdot 19 \pmod{n}$$

$$9398 = \sqrt{23n} + 4; \quad 9398^2 \equiv 59375 = 5^5 \cdot 19 \pmod{n}$$

Hmm. Both have a common "19"

Quadratic Sieve (3)

Factor n = 3837523

Want x,y: $x^2 \equiv y^2, but\ x \neq \pm y \pmod{n}$ → gcd(x-y, n) is a factor

Step 3: Pair x's: try to find two non-congruent perfect squares

Example: $(8077 \cdot 9398)^2 \equiv 2 \cdot 19 \cdot 5^5 \cdot 19 = 2 \cdot 5 \cdot (5^2 \cdot 19)^2$

This is close, but ***all*** factors need to be paired

Recall:

$$8077^2 \equiv 38 = 2 \cdot 19 \pmod{n}$$

$$9398^2 \equiv 59375 = 5^5 \cdot 19 \pmod{n}$$

Quadratic Sieve (3b)

Factor n = 3837523

Want x,y: $x^2 \equiv y^2, but\ x \neq \pm y \pmod{n}$ → gcd(x-y, n) is a factor

Step 3: Pair x's: try to find two non-congruent perfect squares

Example: $(8077 \cdot 9398)^2 \equiv 2 \cdot 19 \cdot 5^5 \cdot 19 = 2 \cdot 5 \cdot (5^2 \cdot 19)^2$

This is close, but ***all*** factors need to be paired

Generate lots of # and experiment until all factors are paired.

$$1964^2 \equiv 3^2 \cdot 13^3 \pmod{n}$$

$$14262^2 \equiv 5^2 \cdot 7^2 \cdot 13 \pmod{n}$$

$$(1954 \cdot 14262)^2 \equiv \left(3 \cdot 5 \cdot 7 \cdot 13^2\right)^2$$

$$1147907^2 \equiv 17745^2$$

So what?

SRCT tells us:
gcd(1147907-17745, n)=1093

Other factor = n/1093=3511

Quadratic Sieve (4)

Factor n = 3837523

Want x,y: $x^2 \equiv y^2, but\ x \neq \pm y \pmod{n}$ → gcd(x-y, n) is a factor

Step 4: Automate finding two non-congruent perfect squares

Example: $(8077 \cdot 9398)^2 \equiv 2 \cdot 19 \cdot 5^5 \cdot 19 = 2 \cdot 5 \cdot (5^2 \cdot 19)^2$

This is close, but ***all*** factors need to be paired

Generate lots of # and experiment until all factors are paired.

To automate this search:

Can write each example as a row in a matrix, where each column is a prime in the number base

Then search for dependencies among rows mod 2.

May need extra rows, since sometimes we get x=+/-y.

My code

Factor n = 3837523

To automate this search:

- Each row in the matrix is a square
- Each column is a prime in the number base
- Search for dependencies among rows mod 2.
- For last one (green)

$$(9398 \cdot 8077 \cdot 3397) \equiv$$
$$-(2^3 \cdot 5^3 \cdot 13 \cdot 19)$$

So we can't use the square root compositeness theorem

```
>> findSquaresOfFactorBaseTerms(3837523, 1, 100, 30)
  1964 = sqrt( 1n +  6), [ 0  2  0  0  0  3  0  0 ]
  3397 = sqrt( 3n +  4), [ 5  0  1  0  0  2  0  0 ]
  3413 = sqrt( 3n + 20), [ 6  0  3  0  0  0  1  0 ]
  3928 = sqrt( 4n + 11), [ 2  2  0  0  0  3  0  0 ]
  5892 = sqrt( 9n + 16), [ 0  4  0  0  0  3  0  0 ]
  6794 = sqrt(12n +  8), [ 7  0  1  0  0  2  0  0 ]
  7856 = sqrt(16n + 21), [ 4  2  0  0  0  3  0  0 ]
  8077 = sqrt(17n +  1), [ 1  0  0  0  0  0  0  1 ]
  8539 = sqrt(19n +  1), [ 4  2  0  0  1  0  0  0 ]
  9207 = sqrt(22n + 19), [ 0  0  0  4  1  1  0  0 ]
  9398 = sqrt(23n +  4), [ 0  0  5  0  0  0  0  1 ]
  9820 = sqrt(25n + 26), [ 0  2  2  0  0  3  0  0 ]
 10191 = sqrt(27n + 12), [ 5  2  1  0  0  2  0  0 ]
 10750 = sqrt(30n + 21), [ 1  0  1  0  2  0  0  2 ]
 12847 = sqrt(43n +  2), [ 4  1  1  1  0  0  0  1 ]
 13588 = sqrt(48n + 16), [ 9  0  1  0  0  2  0  0 ]
 14013 = sqrt(51n + 24), [ 8  1  0  1  2  0  0  0 ]
 14262 = sqrt(53n +  1), [ 0  0  2  2  0  1  0  0 ]
 15425 = sqrt(62n +  1), [ 0  0  0  0  0  1  1  1 ]
 16154 = sqrt(68n +  1), [ 3  0  0  0  0  0  0  1 ]
 16985 = sqrt(75n + 20), [ 5  0  3  0  0  2  0  0 ]
 17078 = sqrt(76n +  1), [ 6  2  0  0  1  0  0  0 ]
 17847 = sqrt(83n +  1), [ 3  0  3  0  0  0  0  0 ]
 18068 = sqrt(85n +  8), [ 0  6  0  0  0  0  0  2 ]
 18796 = sqrt(92n +  7), [ 2  0  5  0  0  0  0  1 ]
 19095 = sqrt(95n +  2), [ 2  0  1  0  1  1  0  1 ]
>>
```

Sum:	0	2	2	2	0	4	0	0
Sum:	8	4	6	0	2	4	0	2
Sum:	6	0	6	0	0	2	0	2

Factoring Summary

1. *Fermat factoring*:
   - Compute $n + 1^2$, $n + 2^2$, $n + 3^2$, until you reach a perfect square, say $r^2 = n + k^2$
   - Then $n = r^2 - k^2 = (r+k)(r-k)$
2. (p-1) algorithm:
   - If (p-1) has only small factors, one can factor n:
   - Compute $b=a^{B!}(\text{mod } n)$, then $d=\gcd(b-1, n)$ is a factor.
   - How to avoid this?
3. Quadratic sieve
   - Generate lots of squares that can be expressed as products of small primes
   - Pairs = linear dependencies (mod 2)
   - Speed? See http://www.crypto-world.com/FactorRecords.html

Discrete logs…

But first, some humor:

Bruce Schneier is a genius in the crypto field, the author of the authoritative book on crypto.

Bruce Schneier writes his books and essays by generating random alphanumeric text of an appropriate length and then decrypting it.

Discrete logs…

…are the basis of the ElGamal cryptosystem

….can be used for digital signatures

Discrete Logs

Given $$\beta = \alpha^{x} \pmod{p}$$

Find x

We denote this as $x = L_{\alpha}(\beta)$

Why is this hard?

Consider this…

- Solve $9=2^x \pmod{11}$
- We denote the answer as $L_2(9)$

- Are there other solutions for x?

- By convention, x is defined to be the minimum of all such.
- It must be < (p-1). Why?

But consider this…

- Solve $2150=3621^x \pmod{p}$ where p=1775754…74581 (100 digits)

- How long will exhaustive search take?
  - Up to p-2 if 3621 is a *primitive root* of n.

- What's a primitive root?

- Please read section 3.7 (1 page) tonight if you haven't

One-way functions

- Take y=f(x)
- If y is easy to find given x, but x is hard to find given y, f is called a *one-way* function.
- Examples:
  - Factoring (easy to multiply, hard to factor)
  - Discrete logs (easy to find powers mod n, even if n is large, but hard to find discrete log)