Cryptography examples using Sage

Christopher Davis

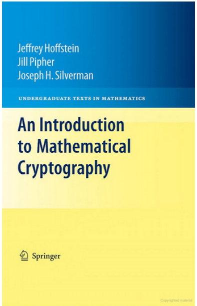
UC Irvine
davis@math.uci.edu
http://math.uci.edu/~davis

Sage Education Days 5
June 21st, 2013

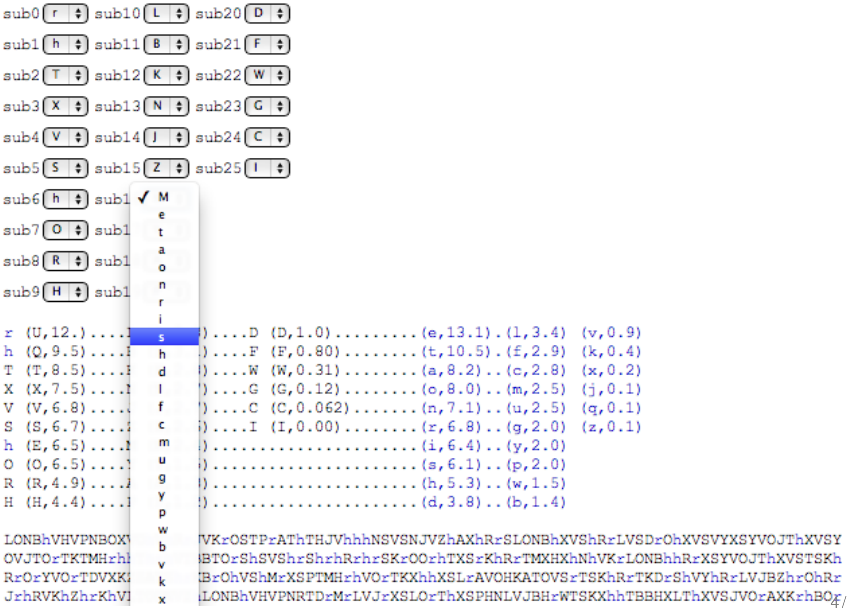
Outline of the talk

About the course

- These materials were designed for UC Irvine Math 173A & Math 173B. Undergraduate cryptography courses.
- No number theory or programming prerequisites.
- Textbook: **An introduction to mathematical crypography** by Hoffstein, Pipher, and Silverman.



Substitution cipher



Computational Complexity Results

x-axis: number of digits of the input
y-axis: time (in seconds) required for the computation

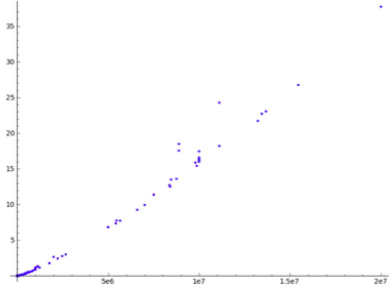


FIGURE 5. GCD

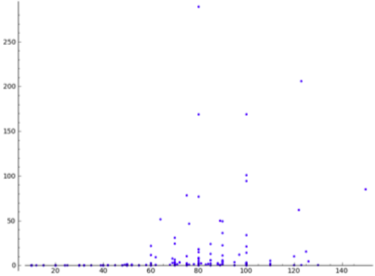


FIGURE 6. Factorization

Students were asked to find numbers which took Sage 1 second to factor, 10 seconds to factor, > 60 seconds to factor. They posted the results on a messageboard, and the data was plotted.

Quadratic sieve

i:	241	242	243	244	245	246	247	248	249	250
pow(i,2,57997):	84	567	1052	1539	2028	2519	3012	3507	4004	4503
2	↓2		↓2		↓2		↓2		↓2	
	42	567	526	1539	1014	2519	1506	3507	2002	4503
3	↓3	↓3		↓3	↓3		↓3	↓3		↓3
	14	189	526	513	338	2519	502	1169	2002	1501
4	↓2		↓2		↓2		↓2		↓2	
	7	189	263	513	169	2519	251	1169	1001	1501
5										
	7	189	263	513	169	2519	251	1169	1001	1501
7	↓7	↓7						↓7	↓7	
	1	27	263	513	169	2519	251	167	143	1501
8										
	1	27	263	513	169	2519	251	167	143	1501
9		↓3		↓3						
	1	9	263	171	169	2519	251	167	143	1501
11						↓11			↓11	
	1	9	263	171	169	229	251	167	13	1501
13					↓13				↓13	
	1	9	263	171	13	229	251	167	1	1501
16										
	1	9	263	171	13	229	251	167	1	1501
17										
	1	9	263	171	13	229	251	167	1	1501
19				↓19						↓19
	1	9	263	9	13	229	251	167	1	79
i	241	242	243	244	245	246	247	248	249	250

Part 1: Private key cryptography

- Relatively small portion of the course: about 3 weeks out of 20 total.
- Materials more “fun”, less mathematically sophisticated.
- Accessible to essentially any audience.

Frequency analysis

- Encryption systems which encrypt English text using substitutions tend to be vulnerable to **frequency analysis.**
- Three important examples:
  - Shift cipher (or Caesar cipher);
  - Substitution cipher;
  - Vigenère cipher.
- Frequency analysis is painful by hand, but easy with Sage.

The Vigenère cryptosystem

- Goal: Encrypt a piece of English text by making substitutions letter-by-letter.
- Shared private key: A list of numbers (shift amounts).

```
vigenere_key = [2,13,9,-1]
```

```
ciphervigenere("hello there, how are you?", vigenere_key)
```

```
'JRUKQGQDTRQNYNADABD'
```

- The first letter is shifted by 2, the fourth letter by -1, the fifth letter by 2 again, etc.

Deciphering Vigenère ciphertext, first step

- First step is to determine the key length. Look for repeated strings.

```
findgaplengths(text,6)
```

```
[25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
130, 140, 140, 140, 140, 140, 140, 140, 160, 295, 295, 295, 320, 320,
320, 320, 320, 320, 320, 320, 320, 320, 320, 380, 490, 495, 620, 660,
660, 735, 840, 840, 840, 840, 885, 885, 885, 885, 1230, 1230, 1230,
1230, 1230, 1230, 1230, 1230]
```

- The function findgaplengths returns the distance separating repeated strings. (In the above image, it looks for repeated strings of length 6.)
- Expectation: the key length should divide many of these gap lengths.

Deciphering Vigenère ciphertext, second step

- Once we have determined key length 5 (say), the second step is to determine the entries in the key.

$$\text{key} = [?, ?, ?, ?, ?]$$

- Separate out every 5th letter and calculate the letter distribution. Then determine which shift amount makes the letter distribution most closely match English.
- In the following, we'll see that shift -9 works best.
- Repeat for the letters in the "1 modulo 5" positions, etc.

Letter distributions

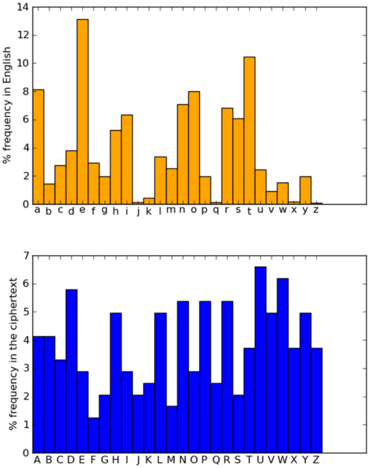


Figure: Letter distributions after picking out every 4th letter

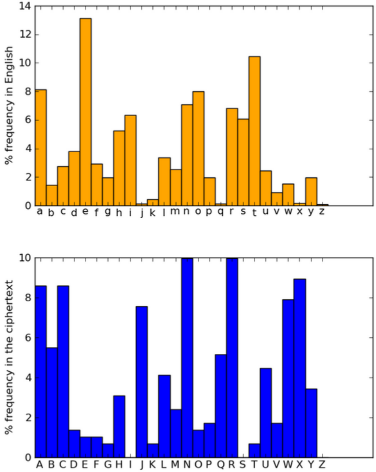


Figure: Letter distributions after picking out every 5th letter

Sample assignments for the Vigenère cipher

1. Some text has been encrypted using a Vigenère cipher with key length 4-8. Decipher it using Sage helper functions.
2. Some (longer) text has been encrypted using a Vigenère cipher with key length < 20. Write a program which will decipher it automatically.

Public key example: Diffie-Hellman key exchange

- Goal: For two people, Alice and Bob, to produce a shared secret number, without ever exchanging any data in private.
- Starting data: A prime p and a generator g for $(\mathbb{Z}/p\mathbb{Z})^\times$.
- Alice has a secret exponent a. Bob has b.
- Alice and Bob publish $g^a \bmod p$ and $g^b \bmod p$.
- Alice and Bob can both compute $g^{ab} \bmod p$ easily, but (we think) nobody else can.

Public key cryptography comments

- Public key cryptography relies on one-to-one functions which are difficult to invert.
- In the Diffie-Hellman example, this function is

$$\begin{aligned} \mathbb{Z}/(p-1)\mathbb{Z} &\to (\mathbb{Z}/p\mathbb{Z})^\times \\ a &\mapsto g^a \bmod p \end{aligned}$$

- "Obvious" benefit to using Sage: can use sizes of numbers for which the functions are genuinely difficult to invert.
- Practice with Sage makes clear to the students that an operation like modular exponentiation is much "easier/faster" than an operation like a discrete logarithm.

Powers of 2 modulo 263

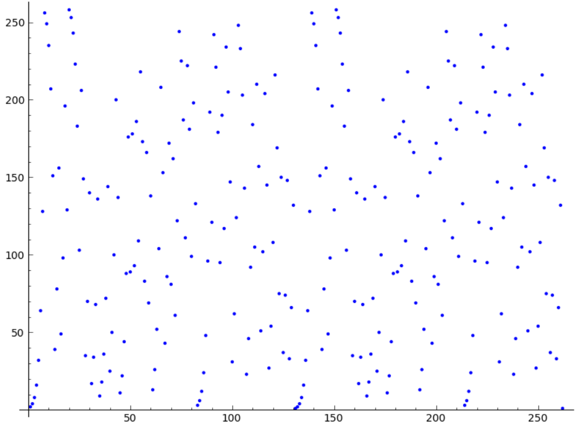


Figure: This modular exponentiation function is so random, it's not surprising it's difficult to invert.

Computation times related to Diffie-Hellman

```
time A = pow(3,a,p)
time output = naive_dlog(3,A,p)
time list1 = baby_step(p,3,A,n)
```

```
Time: CPU 0.00 s, Wall: 0.00 s
1237072
Time: CPU 15.32 s, Wall: 15.32 s
Time: CPU 0.04 s, Wall: 0.04 s
```

Figure: Here p has 7 digits. Computation times for
(1) Modular exponentiation,
(2) trial-and-error discrete log,
(3) a piece of collision algorithm.

```
time A = pow(3,a,p)
time list1 = baby_step(p,3,A,n)
```

```
Time: CPU 0.00 s, Wall: 0.00 s
Time: CPU 78.53 s, Wall: 78.55 s
```

Figure: Here p has 14 digits. The collision algorithm is close to becoming impractical here. In the lab, students were asked to choose p with about 10 more digits.

Three sample Diffie-Hellman assignments

1. Determine ranges where we can't perform the collision algorithm in a reasonable amount of time.
2. With a partner, perform a Diffie-Hellman key exchange using numbers slightly larger than are practical with the collision algorithm.
3. Use the Pohlig-Hellman algorithm to reveal another team's message.

Course summary: What worked best?

- If an assignment would end with a number being revealed, have the students use that number to decipher a block of ciphertext.
- For public key assignments, have students communicate entirely on a public messageboard.
- Have sequential assignments:
  - First assignment: straighforward exchange with a partner using a cryptosystem.
  - Next assignment: stealing another team's exchanged message using an attack on the cryptosystem.

Thank you!

Thank you for your attention!

Any questions?