Lecture 5: Logs and Trees

Menu
- Ron Rivest’s talk
  - Mixnets and voting
  - Public-key cryptography (dynamic programming)
- Trees
- PS1 Grading

MIXes

Encrypted Votes

Decrypted Votes

Random, secret permutation

Security Properties

1. Voters must be able to create votes, but not decrypt them.
2. Observer should be able to verify that output votes correspond to input votes, but not match up votes.

Public-Key Cryptography
- Private procedure: $E$
- Public procedure: $D$
- Identity: $E(D(m)) = D(E(m)) = m$
- Secure: cannot determine $E$ from $D$
- Concept stated by Whitfield Diffie and Martin Hellman in 1976, but didn’t know how to find suitable $E$ and $D$

RSA
The era of “electronic mail” [Potter1977] may soon be upon us; we must ensure that two important properties of the current “paper mail” system are preserved: (a) messages are private, and (b) messages can be signed.

We will not attempt to explain why it works and is (probably) secure in CS216. See links to paper and slides.

Public encryption function

Private encryption function

---

Implementing RSA

```python
def RSAencrypt(M, e, n):
    if e == 0:
        return 1
    else:
        return (M * RSAencrypt(M, e - 1, n)) % n
```

Note: this actually "works" in Python even though RSA needs 100+-digit values (but not in Java) because integers are not limited to a fixed size. We'll consider number representations later in the class.

---

Result

```python
... File "C:\cs216\workspace\ps2\RSA.py", line 17, in RSAencrypt
    return (M * RSAencrypt(M, e - 1, n)) % n
  File "C:\cs216\workspace\ps2\RSA.py", line 17, in RSAencrypt
    return (M * RSAencrypt(M, e - 1, n)) % n
  File "C:\cs216\workspace\ps2\RSA.py", line 17, in RSAencrypt
    return (M * RSAencrypt(M, e - 1, n)) % n
RuntimeError: maximum recursion depth exceeded
```

---

Analyzing RSA

```python
def RSAencrypt(M, e, n):
    if e == 0:
        return 1
    else:
        return (M * RSAencrypt(M, e - 1, n)) % n
```

How many recursive calls?
The value of the $e$ parameter (scales as $O(2^e)$ size of $e$)

Can we use dynamic programming (and math) to make this faster?

---

Fast Exponentiation

```python
def square(x): return x * x
def RSAencrypt(M, e, n):
    if e == 0:
        return 1
    elif e % 2 == 0:
        return square(RSAencrypt(M, e/2, n)) % n
    else:
        return (M * RSAencrypt(M, e - 1, n)) % n
```

Multiplication is associative

---

Fast Exponentiation

```python
def square(x): return x * x
def RSAencrypt(M, e, n):
    if e == 0:
        return 1
    elif e % 2 == 0:
        return square(RSAencrypt(M, e/2, n)) % n
    else:
        return (M * RSAencrypt(M, e - 1, n)) % n
```
Analysis

```
def square (x): return x * x

def RSAencrypt (M, e, n):
    if e == 0: return 1
    elif e % 2 == 0:
        return square(RSAencrypt (M, e/2, n)) % n
    else:
        return (M * RSAencrypt (M, e - 1, n)) % n
```

• How many recursive calls?
  Worst case:  $e = 2^q - 1$
  $2q$ calls $\in \Theta(\log_2 e)$

Worst case:  $e = 2^q - 1$
  $2q$ calls $\in \Theta(\log_2 e)$

Logarithms Review

• Logarithm is inverse of exponential
  $x = \log_b b^x$

• Bases
  - Is $\log_2 f(x) \in O(\log_3 f(x))$?
  - Is $\log_2 f(x) \in \Omega(\log_3 f(x))$?

Changing Bases

\[
\begin{align*}
x &= b^{\log_b x} \\
\log_a x &= \log_a (b^{\log_b x}) \\
\log_a x &= \log_a b \times \log_a (\log_b x) \\
1 &= \log_a b / \log_a x \times \log_b x
\end{align*}
\]

\[
\log_b x = \log_a x / \log_a b
\]

So, within $O$, $\Theta$, $\Omega$ the base doesn’t matter

Logs and Trees

• Many tree algorithms have running time $\in \log(N)$ where $N$ is the number of nodes in the tree, since for a well balanced tree
  \[
  N = b^{h+1} + 1 \\
h \sim \log_b(N)
  \]

Implementing RSA

```
def square (x): return x * x

def RSAencrypt (M, e, n):
    if e == 0: return 1
    elif e % 2 == 0:
        return square(RSAencrypt (M, e/2, n)) % n
    else:
        return (M * RSAencrypt (M, e - 1, n)) % n
```

• Recursive calls $\in \Theta(\log_2 e)$
• Running time $\in \Theta(\log_2 e)$ if multiplication time is $O(1)$

Note that this cannot really be true!

Public-Key Applications: Privacy

- Alice encrypts message to Bob using Bob’s Private Key
- Only Bob knows Bob’s Private Key $\Rightarrow$ only Bob can decrypt message
Signatures

- Bob knows it was from Alice, since only Alice knows Alice’s Private Key
- Non-repudiation: Alice can’t deny signing message (except by claiming her key was stolen!)
- Integrity: Bob can’t change message (doesn’t know Alice’s Private Key)

Encrypt

Decrypt

Plaintext

Signed Message

C1 = E_{KUM} ["Kerry"]

M = E_{KR_M} [C]

M1

M2

M3

M4

C1

C2

C3

C4

Opps...doesn’t work: anyone can use public key to compute \( E_{KUM} [M] \) for outputs and compare to inputs.

Random, secret permutation

Mux keys: \( K_{U_M}, K_{R_M} \)

\[ C = E_{KUG} ["Badnarik" \| R_1] \]

Does publishing \( R_1 \) help?

\[ \text{Input} = E_{KUR} [E_{KUD} [E_{KUG} [v \| R_1] R_2] R_3] \]

\[ \text{Output} = E_{KED} [E_{KUG} [v \| R_1] R_2] \]

If \( R \) reveals \( j \) and \( R_1 \), \( D \) can check \( E_{KUR} [\text{Output}, \| R_1] = \text{Input} \)
Catching Cheaters

- Probability a mux can cheat on $k$ votes without getting caught = $\frac{1}{2^k}$
- Probability a voter's vote is revealed to eavesdropper
  - $m$ muxes
  - Note: unaudited votes only be one of $n/2$ possible outputs!
- If muxes collude, all bets are off

Voting Caveat

- Real problems with voting have very little to do with cryptography or mixes...

Trees

Tree Node Operations

- getLeftChild (Node)
- getRightChild (Node)
- getInfo (Node)

```python
def isLeaf(self):
    return self.getLeftChild() == None and self.getRightChild() == None
```

Calculating Height

```python
def height(self):
    if self.isLeaf():
        return 0
    else:
        return 1 + max(self.getLeftChild().height(), self.getRightChild().height())
```

Height of a Node: length of the longest path from that node to a leaf

Analyzing Height

- What is the asymptotic running time or our height procedure?
```python
def height(self):
    if self.isLeaf():
        return 0
    else:
        return 1 + max(self.getLeftChild().height(), self.getRightChild().height())
```
Tree Preorder Traversal

Preorder Traversal

```python
def preorder(t):
    print t.info()
    for c in t.children():
        c.preorder()
```

- $N$ is number of nodes in $t$
- Running time: $\Theta(N)$
- Space use: worst case: $O(N)$
  well-balanced case: $O(\log N)$

PS1

Returned in section today

CS216 PS Grading Scale

- Gold Star – Excellent Work. You got everything I wanted on this PS.
- Green Star – Better than good work
- Blue Star – Good Work. You got most things on this PS, but some answers could be better.
- Silver Star – Some problems. Make sure you understand the comments.
- Red Star – Serious problems and lack of effort.

PS1 Average: ★★

No upper limit

- Double Gold Star: exceptional work! Better than I expected anyone would do.
- Triple Gold Star: Better than I thought possible
- Quadruple Gold Star: You have broken important new ground in CS which should be published in a major journal! (e.g., invented an alignment algorithm better than BLAST)
- Quintuple Gold Star: You deserve a Turing Award! (find an $O(n^2)$ solution to finding optimal phylogenies)

Philosophy

“This generation of students got into Harvard by doing exactly and precisely what teacher wants. If teacher is vague about what he [sic] wants, they work a lot harder to figure out what they want and whether or not it is good. The vaguer the directions, the more likely the opportunity for serendipity to happen. It drives them nuts!”

Harvard Professor John Stilgoe
(on 60 Minutes, 4 January 2004)
Charge

• Today and tomorrow:
  – Sections in Thorton D classrooms
• Wednesday: PS2 is Due!