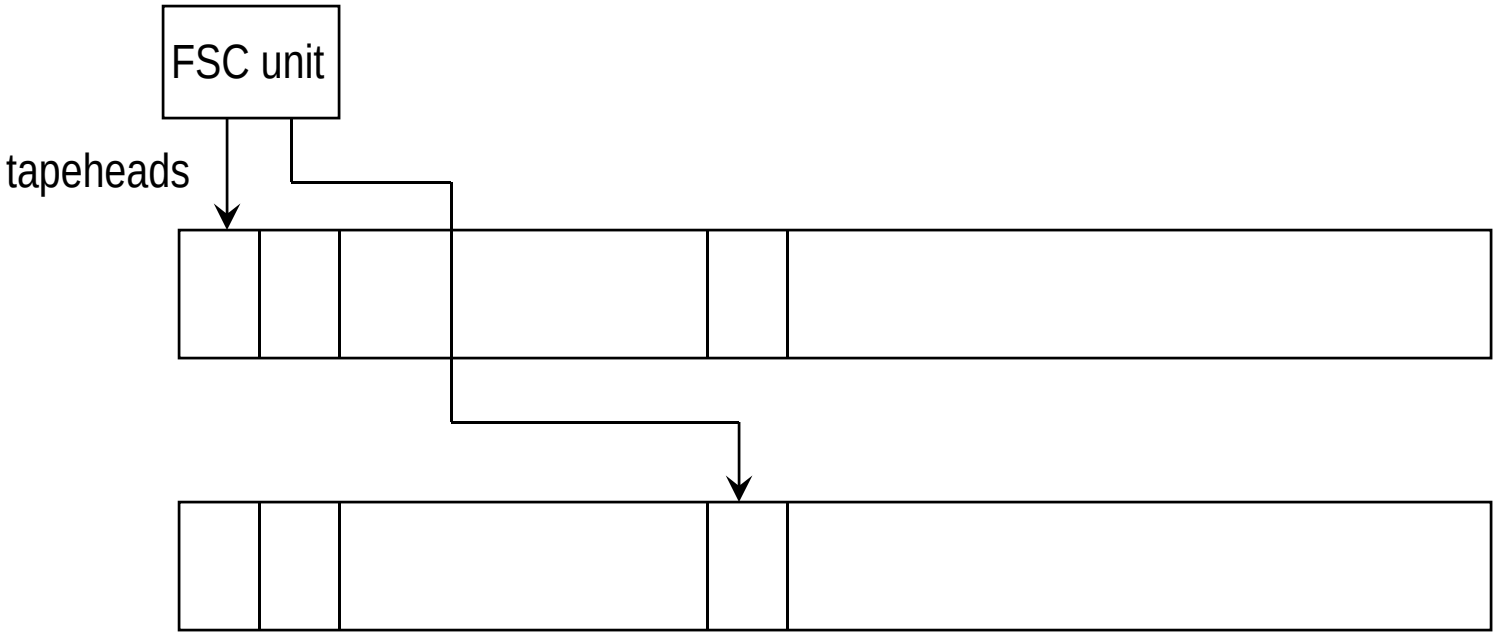
CIS 5371 Cryptography

4. Computational Complexity

Turing Machines

- A finite *state control unit*
- $k \geq 1$ *tapes* and *read* or *write tapeheads*



Deterministic Polynomial Time

- ## Class P

The class of languages L such that:

any $x \in L$ can be recognized by a TM in time bounded by a polynomial p(n), where n is the length of x.

A language recognition problem is a *decisional* problem.

Class P can also be used to capture *computational* problems.

- ## Polynomial-time computational problems

These must be computed by a TM in polynomial-time.

The Euclidean Algorithm

Input: integers $a > b \geq 0$

Output: $gcd(a,b)$

1. $i \leftarrow 0$; $r_{-1} \leftarrow a$; $r_0 \leftarrow b$;
2. while ($r_i \neq 0$) do
   a) $r_{i+1} \leftarrow_. r_{i-1} (\text{mod } r_i)$;
   b) $i \leftarrow i+1$
3. return(r_{i-1})

An example

$a = 60, b = 9$

r_{i-1}	r_i
60	9
9	6
6	3
3	0

$3 = gcd(60,9)$

The Extended Euclidean Algorithm

Input: integers $a > b \geq 0$

Output: integers λ, μ satisfying $a\lambda + b\mu = gcd(a,b)$

1. $i \leftarrow 0$; $r_{-1} \leftarrow a$; $r_0 \leftarrow b$; $q \leftarrow 0$
   $\lambda_{-1} \leftarrow 1$; $\mu_{-1} \leftarrow 0$; $\lambda_0 \leftarrow 0$; $\mu_0 \leftarrow 1$
2. while ($r_i = a\lambda_i + b\mu_i \neq 0$) do
   a) $q = r_{i-1} \div r_i$
   b) $\lambda_{i+1} \leftarrow_. \lambda_{i-1} - q\lambda_i$; $\mu_{i+1} \leftarrow_. \mu_{i-1} - q\mu_i$
   c) $r_{i+1} \leftarrow_. r_{i-1} \pmod{r_i}$;
   d) $i \leftarrow i+1$
3. return((λ_i , μ_i))

An example

$a = 60, b = 9$

i	r	q	λ	μ
-1	60		1	0
0	9	6	0	1
1	6	1	1	-6
2	3	2	-1	7
3	0	×	×	×

$3 = 60*(-1) + 9*(7)$

Modular arithmetic

- Some exercises in modular addition multiplication and exponentiation.
- The computational complexity of modular addition and multiplication
- The computational complexity of modular exponentiation: the square & multiply algorithm

Modular Exponentiation

Input: integers $x > 0$, $y > 0$, $n > 0$.

Output: $x^y \pmod{n}$

mod_exp(x,y,n).

1. if $y = 0$ return (1) ;
2. if $y = 0 \pmod 2$ return (mod_exp($x^2 \pmod n$), $y \div 2$, n) ;
3. return ($x \cdot$ mod_exp($x^2 \pmod n$), $y \div 2$, n).

Probabilistic Polynomial Time

- A *probabilistic* Turing Machine (TM) is a non-deterministic TM with bounded error (<1/3).

- **Class PP**

  The class of languages L such that:

  - *any $x \in L$ can be recognized by a probabilistic TM in time bounded by a polynomial $p(n)$, where n is the length of x.*

PP (Monte Carlo)
Always fast, probably correct

$L \in$ PP (Monte Carlo):

if there exists a randomized algorithm A such that for any instance I:

$$\text{Prob}\,[\, A \text{ recognizes } I \mid I \in L \,] = 1$$

and

$$\text{Prob}\,[\, A \text{ recognizes } I \mid I \notin L \,] \leq \delta,$$

where δ is a constant in the interval $(0,1/2)$.

PP Monte Carlo

Input : p, a positive number
Output : Yes if p is prime, No otherwise.

1. Repeat $\log_2 p$ times :
   a) Pick a random number x in $(1, p\text{-}1)$;
   b) If $\gcd(x,p) > 1$ or if $x^{(p-1)/2} \neq \pm 1 \pmod{p}$ return (No)
   end of repeat ;
2. If (test in 1.b never shows –1) return (No)
3. Return (Yes)

Primality testing

This test holds because

- Fermat's little theorem: if p is a prime then:
  - For any $0 < x < p$: $x^{p-1} \equiv 1 \pmod{p}$
  - There are only two quadratic residues of 1: +1 and -1.
- If n is a composite number then there are at least 4 quadratic residues of 1 modulo n.
  - Consequently we may have $x^{(p-1)/2} \neq \pm 1 \pmod{p}$

PP (Las Vegas)
Probably fast, always correct

$L \in$ PP (Las Vegas):

if there exists a randomized algorithm A such that for any instance I:

$$\text{Prob}\,[\, A \text{ recognizes } I \mid I \in L \,] \geq \varepsilon$$

and

$$\text{Prob}\,[\, A \text{ recognizes } I \mid I \notin L \,] = 0,$$

where ε is a constant in the interval $(1/2,1)$.

PP Las Vegas

Input : p, an odd positive number

$q_1, q_2, \ldots, q_k$: all prime factors of p-1

Output : Yes if p is prime, No otherwise.

No decision with certain probability of error

1. Pick $g \in_U [2,p-1]$
2. For (i=1, i++, k) do
   if $g^{(p-1)/q_i} = 1 \pmod{p}$ output NO_DECISION and terminate;
3. if $g^{p-1} \neq 1 \pmod{p}$ output NO and terminate;
4. Output YES and terminate.

Primality testing

This test holds because p is a prime iff there exists some $g \in [2,p-1]$

- $g^{p-1} \equiv 1 \pmod{p}$,
- $g^{(p-1)/q_i} \neq 1 \pmod{p}$ for all factors of p-1.

Non-deterministic Polynomial Time

- Class NP
- These are recognized by non-deterministic Turing Machine in polynomial time.
- Instances of NP problems have witnesses which can be found by random guessing: given a *witness*, every language (decisional problem) is recognizable in polynomial time.

Class NP -- examples

- *Square-freeness*
- *Quadratic residuocity*
- Other examples.

Clearly we have:

$$P \subseteq NP \text{ problems}$$

Class NP-Complete

A general discussion on NP-*completeness.*