Midterm

Martin Kozeny, David Kalivoda
CSCI 4501: Intro Cryptography
Spring 2011 University of New Orleans

March 21, 2011

1. Sketch out in pseudocode the 'memory efficient' (that is, a version that does not require it to remember all of the steps) version of the extended Euclidean algorithm discussed in class.

```
func euclidean(var Zn, var n)

  //initialization of variables to start looping properly
  var prevA=0;
  var prevB=1;
  var prevprevA=1;
  var prevprevB=0;

  //loop computing extended euclidean algorithm, the result is in prevA and prevB
  //loop uses continous dividing Zn by n and shifting parameters from each round
  loop( (Zn%n) != 0){
   a = Zn/n;
   b = Zn%n;

   //we need this temp variables for swapping coeficients
   //from prevprev to prev (preparation for next round)
   tempA = prevA;
   tempB = prevB;

   //equations similar to those when we do in hand algorithm
   prevA = prevprevA - a*prevA;
   prevB = prevprevB - a*prevB;

   prevprevA = tempA;
   prevprevB = tempB;

   Zn = n;
   n  = b;
  }//end of loop
  //inverse condition
  if (n!=1) Inverse does not exist;

  //in return container are the multiplicative constantes of eucledian
  //expansion
  //we usually need only the prevB constant
  return prevB;
```

2. Sketch out in pseudocode the algorithm which allows us to calculate $x^y \mod m$ for very large values of x, y, m.

```
func power(var x, var y, var Zn)
  y = toBinaryNumber(y);
  //function convertToExpression() coverts input binary number
  //such that instead of ones puts 'X' and into spaces between bits puts 'S'
  //e.g. 23 = 10101 in binary -> XSSXSSX
  var powerExpression = convertToExpression(y);
  //now we are going to do powering x to y
  var result = x;
  //this looping starts from most significant char in string of powerExpression
  loop through bits of powerExpression{
    if 'X'
      multiply result from previous loop by x mod Zn
    else if 'S'
      square result from previous loop mod Zn
  }//end of loop
  //now x^y is in result
  return result
```

Note: The most expensive operation in this function is squaring and moduling, which we can compute manually.

3. Do the following in $GF(16) = ((Z/_2)[x])/(x^4 + x + 1)$

Fully simplify your answer.

(a) Compute $(x^2 + 1)(x^3 + x^2 + 1)$.

$$(x^2 + 1)(x^3 + x^2 + 1) = x^5 + x^4 + x^3 + 1$$
$$(x^5 + x^4 + x^3 + 1)/(x^4 + x + 1) = x + 1,\ remainder\ x^3 + x^2$$
$$(x^2 + 1)(x^3 + x^2 + 1) \equiv x^3 + x^2\ in\ GF(16) = ((Z/_2)[x])/(x^4 + x + 1)$$

(b) Find the inverse of $x^2 + x + 1$.

$$(x^4 + x + 1)/(x^2 + x + 1) = x^2 + x,\ remainder\ 1$$
$$(x^4 + x + 1) = (x^2 + x)(x^2 + x + 1) + 1$$
$$1 = (x^2 + x)(x^2 + x + 1) + 1(x^4 + x + 1)$$

Multiplicative inverse of $(x^2 + x + 1)$ in $GF(16) = ((Z/_2)[x])/(x^4 + x + 1)$ is $(x^2 + x)$.

4. Let, $p = 5$, $q = 7$, and $e = 17$ for RSA encryption. Set $x = 13$. Encrypt x using RSA. Then find d and decrypt back to the original message.

$$n = pq = 5 \cdot 7 = 35$$
$$\phi(35) = (5 - 1)(7 - 1) = 24$$
$$17 = (10001)_2 = XSSSSX$$

For computation 13^{17} in $\mathbb{Z}_{35}$ we used algorithm shown in second question.

$$13^{17} \equiv 13 \mod 35$$

Encrypted message is 13. For decrypting we have to find inverse $d = e^{-1}$, which is 17^{-1} in $\mathbb{Z}_{24}$.

$$24 = 1 \cdot 17 + 7$$

$$17 = 2 \cdot 7 + 3$$
$$7 = 2 \cdot 3 + 1$$

$$7 = 24 - 1 \cdot 17$$
$$3 = 17 - 2 \cdot 24 + 2 \cdot 17 = 3 \cdot 17 - 2 \cdot 24$$
$$1 = 24 - 1 \cdot 17 - 6 \cdot 17 + 4 \cdot 24 = 5 \cdot 24 - 7 \cdot 17$$

$$d = -7 \equiv 17 \; in \; \mathbb{Z}_{24}$$

Because $d = e$, encrypting is the same as decrypting. We calculate $13^{17} \equiv 13$ in $\mathbb{Z}_{35}$ as shown before.

5. Let p be a prime. What is $(\prod_{i=1}^{p-1} i) \mod p$? Fully justify (i.e., prove) your answer.

   $(\prod_{i=1}^{p-1} i) \mod p = 1 \cdot 2 \ldots (p-1)$

   We know, that all elements in $\mathbb{Z}_p$ have inverses. We can see, that 1 has inverse onto itself. We try to find, if any other elements of group $\mathbb{Z}_p$ have the same quality.

$$(p-n)(p-n) \equiv 1 \mod p$$

$$, \text{where } n \in \{1, 2, \ldots, (p-1)\}$$

$$n^2 - 2pn + p^2 - 1 \equiv 0 \mod p$$
$$(n - (p-1))(n - (p+1)) \equiv 0 \mod p$$
$$n_1 = p + 1 \equiv 1 \mod p$$
$$n_2 \equiv (p-1) \mod p$$

We see that only $(p - n_1) \equiv (p-1) \mod p$ and $(p - n_2) = (p - (p-1)) \equiv 1 \mod p$ have inverse onto itself. We might find **pairs of inverses** for the rest elements in group. Each element multiplied by its inverse is one, and therefore:

$$(\prod_{i=1}^{p-1} i) \mod p = 1 \cdot \underbrace{2 \cdot 3 \ldots (p-2)}_{1} \cdot (p-1) = (p-1)$$