CS255: Cryptography and Computer Security **Winter 2000**

Very basic number theory fact sheet

Part I: Arithmetic modulo primes

Basic stuff

1. We are dealing with primes p on the order of 300 digits long, (1024 bits).

2. For a prime p let $\mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$.
   Elements of $\mathbb{Z}_p$ can be added modulo p and multiplied modulo p.

3. Fermat's theorem: for any $g \neq 0 \bmod p$ we have: $g^{p-1} = 1 \bmod p$.
   Example: $3^4 \bmod 5 = 81 \bmod 5 = 1$

4. The *inverse* of $x \in \mathbb{Z}_p$ is an element a satisfying $a \cdot x = 1 \bmod p$.
   The inverse of x modulo p is denoted by x^{-1}.
   Example: 1. $3^{-1} \bmod 5 = 2$ since $2 \cdot 3 = 1 \bmod 5$.
   2. $2^{-1} \bmod p = \frac{p+1}{2}$.

5. All elements $x \in \mathbb{Z}_p$ except for $x = 0$ are invertible.
   Simple inversion algorithm: $x^{-1} = x^{p-2} \bmod p$.
   Indeed, $x^{p-2} \cdot x = x^{p-1} = 1 \bmod p$.

6. Denote by $\mathbb{Z}_p^*$ the set of invertible elements in $\mathbb{Z}_p$. Hence, $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$.

7. We now have algorithm for solving linear equations: $a \cdot x = b \bmod p$.
   Solution: $x = b \cdot a^{-1} = b \cdot a^{p-2} \bmod p$.
   What about an algorithm for solving quadratic equations?

Structure of $\mathbb{Z}_p^*$

1. $\mathbb{Z}_p^*$ is a *cyclic group*.
   In other words, there exists $g \in \mathbb{Z}_p^*$ such that $\mathbb{Z}_p^* = \{1, g, g^2, g^3, \ldots, g^{p-2}\}$.
   Such a g is called a *generator* of $\mathbb{Z}_p^*$.
   Example: in $\mathbb{Z}_7^*$: $\langle 3 \rangle = \{1, 3, 3^2, 3^3, 3^4, 3^5, 3^6\} = \{1, 3, 2, 6, 4, 5\} \pmod 7 = \mathbb{Z}_7^*$.

2. Not every element of $\mathbb{Z}_p^*$ is a generator.
   Example: in $\mathbb{Z}_7^*$ we have $\langle 2 \rangle = \{1, 2, 4\} \neq \mathbb{Z}_7^*$.

3. The *order* of $g \in \mathbb{Z}_p^*$ is the smallest positive integer a such that $g^a = 1 \bmod p$.
   The order of $g \in \mathbb{Z}_p^*$ is denoted $\mathrm{ord}_p(g)$.
   Example: $\mathrm{ord}_7(3) = 6$ and $\mathrm{ord}_7(2) = 3$.

4. Lagrange's theorem: for all $g \in \mathbb{Z}_p^*$ we have that $\mathrm{ord}_p(g)$ divides $p-1$.

5. If the factorization of $p-1$ is known then there is a simple and efficient algorithm to determine $\text{ord}_p(g)$ for any $g \in \mathbb{Z}_p^*$.

Quadratic residues

1. The *square root* of $x \in \mathbb{Z}_p$ is a number $y \in \mathbb{Z}_p$ such that $y^2 = x \bmod p$.
   Example: 1. $\sqrt{2} \bmod 7 = 3$ since $3^2 = 2 \bmod 7$.
   2. $\sqrt{3} \bmod 7$ does not exist.

2. An element $x \in \mathbb{Z}_p^*$ is called a *Quadratic Residue* (QR for short) if it has a square root in $\mathbb{Z}_p$.

3. How many square roots does $x \in \mathbb{Z}_p$ have?
   If $x^2 = y^2 \bmod p$ then $0 = x^2 - y^2 = (x-y)(x+y) \bmod p$.
   Since $\mathbb{Z}_p$ is an "integral domain" we know that $x = y$ or $x = -y \bmod p$.
   Hence, elements in $\mathbb{Z}_p$ have either zero square roots or two square roots.
   If a is the square root of x then $-a$ is also a square root of x modulo p.

4. Euler's theorem: $x \in \mathbb{Z}_p$ is a QR if and only if $x^{(p-1)/2} = 1 \bmod p$.
   Example: $2^{(7-1)/2} = 1 \bmod 7$ but $3^{(7-1)/2} = -1 \bmod 7$.

5. Let $g \in \mathbb{Z}_p^*$. Then $a = g^{(p-1)/2}$ is a square root of 1. Indeed, $a^2 = g^{p-1} = 1 \bmod p$.
   Square roots of 1 modulo p are 1 and -1.
   Hence, for $g \in \mathbb{Z}_p^*$ we know that $g^{(p-1)/2}$ is 1 or -1.

6. Legendre symbol: for $x \in Z_p$ define $\left(\frac{x}{p}\right) = \begin{cases} 1 & \text{if } x \text{ is a QR in } \mathbb{Z}_p \\ -1 & \text{if } x \text{ is not a QR in } \mathbb{Z}_p \\ 0 & \text{if } x = 0 \bmod p \end{cases}$.

7. By Euler's theorem we know that $\left(\frac{x}{p}\right) = x^{(p-1)/2} \bmod p$.
   $\Longrightarrow$ the Legendre symbol can be efficiently computed.

8. Easy fact: let g be a generator of $\mathbb{Z}_p^*$. Let $x = g^r$ for some integer r.
   Then x is a QR in $\mathbb{Z}_p$ if and only if r is even.
   $\Longrightarrow$ **the Legendre symbol reveals the parity of** r.

9. Since $x = g^r$ is a QR if and only if r is even it follows that exactly half the elements of $\mathbb{Z}_p$ are QR's.

10. When $p = 3 \bmod 4$ computing square roots of $x \in \mathbb{Z}_p$ is easy.
    Simply compute $a = x^{(p+1)/4} \bmod p$.
    $a = \sqrt{x}$ since $a^2 = x^{(p+1)/2} = x \cdot x^{(p-1)/2} = x \cdot 1 = x \pmod{p}$.

11. When $p = 1 \bmod 4$ computing square roots in $\mathbb{Z}_p$ is possible but somewhat more complicated (randomized algorithm).

12. We now have an algorithm for solving quaratic equations in $\mathbb{Z}_p$.
    We know that if a solution to $ax^2 + bx + c = 0 \bmod p$ exists then it is given by:

    $$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \pmod{p}$$

    Hence, the equation has a solution in $\mathbb{Z}_p$ if and only if $\Delta = b^2 - 4ac$ is a QR in $\mathbb{Z}_p$. Using our algorithm for taking square roots in $\mathbb{Z}_p$ we can find $\sqrt{\Delta} \bmod p$ and recover x_1 and x_2.

13. What about cubic equations in $\mathbb{Z}_p$? There exists an efficient randomized algorithm that solves any equation of degree d in time polynomial in d.

Computing in $\mathbb{Z}_p$

1. Since p is a huge prime (e.g. 1024 bits long) it cannot be stored in a single register.
2. Elements of $\mathbb{Z}_p$ are stored in buckets where each bucket is 32 or 64 bits long depending on the processor's chip size.
3. Adding two elements $x, y \in \mathbb{Z}_p$ can be done in linear time in the *length* of p.
4. Multiplying two elements $x, y \in \mathbb{Z}_p$ can be done in quadratic time in the *length* of p. If p is n bits long, more clever (and practical) algorithms work in time $O(n^{1.7})$ (rather than $O(n^2)$).
5. Inverting an element $x \in \mathbb{Z}_p$ can be done in quadratic time in the length of p.
6. Using the repeated squaring algorithm, $x^r \bmod p$ can be computed in time $(\log_2 r)O(n^2)$ where p is n bits long. Note, the algorithm takes linear time in the length of r.

Summary

Let p be a 1024 bit prime. Easy problems in $\mathbb{Z}_p$:

1. Generating a random element. Adding and multiplying elements.
2. Computing $g^r \bmod p$ is easy even if r is very large.
3. Inverting an element. Solving linear systems.
4. Testing if an element is a QR and computing its square root if it is a QR.
5. Solving polynomial equations of degree d can be done in polynomial time in d.

Problems that are believed to be hard in $\mathbb{Z}_p$:

1. Let g be a generator of $\mathbb{Z}_p^*$. Given $x \in \mathbb{Z}_p^*$ find an r such that $x = g^r \bmod p$. This is known as the *discrete log problem*.

2. Let g be a generator of $\mathbb{Z}_p^*$. Given $x, y \in \mathbb{Z}_p^*$ where $x = g^{r_1}$ and $y = g^{r_2}$. Find $z = g^{r_1 r_2}$. This is known as the *Diffie-Hellman problem*.

3. Finding roots of sparse polynomials of high degree.
   For example finding a root of: $x^{(2^{500})} + 7 \cdot x^{(2^{301})} + 11 \cdot x^{(2^{157})} + x + 17 = 0 \bmod p$.