CS60082/CS60094 Computational Number Theory, Spring 2010–11

End-Semester Test

Maximum marks: 55 Date: April 2011 Duration: 3 hours

Roll no: _____ **Name:** _____

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions.*]

1. Represent $\mathbb{F}_{27} = \mathbb{F}_{3^3}$ as $\mathbb{F}_3(\theta)$, where $\theta^3 + 2\theta + 1 = 0$. Let $\alpha = \theta^2 + 2$.

(a) Determine whether α is a primitive element of $\mathbb{F}_{27}$. **(5)**

Solution We have $27 - 1 = 2 \times 13$. We compute α^2 and $\alpha^{13} = \alpha \times \alpha^4 \times \alpha^8$.

$$\begin{aligned}
\alpha^2 &= \theta^4 + \theta^2 + 1 = \theta(\theta^3 + 2\theta + 1) + 2\theta^2 + 2\theta + 1 = 2\theta^2 + 2\theta + 1, \\
\alpha^4 &= \theta^4 + \theta^2 + 1 + 2\theta^3 + \theta^2 + \theta = \theta^4 + 2\theta^3 + 2\theta^2 + \theta + 1 \\
&= \theta(\theta^3 + 2\theta + 1) + 2\theta^3 + 1 = 2(\theta^3 + 2\theta + 1) + 2\theta + 2 = 2\theta + 2, \\
\alpha^8 &= \theta^2 + 2\theta + 1.
\end{aligned}$$

But then $\alpha^{13} = (\theta^2 + 2)(2\theta + 2)(\theta^2 + 2\theta + 1) = 2(\theta^2 + 2)(\theta + 1)(\theta^2 - \theta + 1) = 2(\theta^2 + 2)(\theta^3 + 1) = 2\theta(\theta^2 + 2) = 2(\theta^3 + 2\theta) = -2 = 1$. Since $\alpha^{13} = 1$, we conclude that α is not a primitive element.

(b) Determine whether α is a normal element of $\mathbb{F}_{27}$. **(5)**

Solution We have

$$\begin{aligned}
\alpha &= \theta^2 + 2 \\
\alpha^3 &= \alpha \times \alpha^2 = (\theta^2 + 2)(2\theta^2 + 2\theta + 1) \\
&= 2\theta^4 + 2\theta^3 + \theta^2 + \theta^2 + \theta + 2 \\
&= 2\theta^4 + 2\theta^3 + 2\theta^2 + \theta + 2 \\
&= 2\theta(\theta^3 + 2\theta + 1) + 2\theta^3 + \theta^2 + 2\theta + 2 \\
&= 2(\theta^3 + 2\theta + 1) + \theta^2 + \theta = \theta^2 + \theta, \\
\alpha^9 &= \alpha \times \alpha^8 = (\theta^2 + 2)(\theta^2 + 2\theta + 1) \\
&= (\theta - 1)(\theta + 1)(\theta^2 - \theta + 1) = (\theta + 2)(\theta^3 + 1) \\
&= \theta(\theta + 2) = \theta^2 + 2\theta.
\end{aligned}$$

Therefore,

$$\begin{pmatrix} \alpha \\ \alpha^3 \\ \alpha^9 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \theta \\ \theta^2 \end{pmatrix}$$

The determinant of the transformation matrix is $2 \times (1 - 2) \equiv 1 \not\equiv 0 \pmod 3$. Therefore, α is a normal element of $\mathbb{F}_{27}$.

2. Let s and t be bit lengths with $s > t$. Your task is to find a random s-bit prime p for which $p-1$ has a prime divisor of bit length t.

(a) Describe an *efficient* algorithm to compute such a prime p. **(5)**

Solution The following algorithm generates a random s-bit prime p with a random t-bit prime divisor q of $p-1$.

1. Generate a random t-bit integer q.
2. If q is not prime, go to Step 1.
3. Compute the bounds $B_1 = \lceil (2^{s-1}-1)/q \rceil$ and $B_2 = \lfloor (2^s-2)/q \rfloor$.
4. Generate a random integer a in the interval $B_1 \leqslant a \leqslant B_2$.
5. Set $p = aq + 1$.
6. If p is not prime, go to Step 4.
7. Return p (and q, if needed).

(b) Express the expected running time of your algorithm in terms of the bit lengths s and t. **(5)**

Solution The running time of the above algorithm is dominated by the primality tests in Steps 2 and 6. Let a primality test on an l-bit integer run in time $O(l^k)$ for some constant k. By the prime number theorem, the probability for q to be prime (Step 2) is $\Theta(t)$, and the probability for p to be prime (Step 6) is $\Theta(s)$. That is, we expect to obtain a prime q after trying $\Theta(t)$ random values. Moreover, we expect to obtain a prime p after trying $\Theta(s)$ multipliers a. It, therefore, follows that the expected running time of the above algorithm is $O(s^{k+1} + t^{k+1})$, that is, $O(s^{k+1})$ (since $t < s$). For the Miller-Rabin test, $k = 3$, so this running time is $O(s^4)$. For the AKS test, we can take $k = 7.5$, so the running time becomes $O(s^{8.5})$.

3. [*Pocklington primality test*] Let n be a positive odd integer whose primality is to be checked. Write $n-1=uv$, where the complete prime factorization of u is known, whereas v is composite with no known factors. (The case $v=1$ is also allowed.) Suppose also that for some integer a, we have $a^{n-1}\equiv 1 \pmod{n}$, whereas $\gcd(a^{(n-1)/q}-1,n)=1$ for all prime divisors q of u.

(a) Prove that every prime factor p of n satisfies $p\equiv 1 \pmod{u}$. (**Hint:** First, show that $u \mid \operatorname{ord}_p(a)$.) **(5)**

Solution Since $a^{n-1}\equiv 1 \pmod{n}$, we evidently have $a^{n-1}\equiv 1 \pmod{p}$. Moreover, since $p \mid n$, we have $\gcd(a^{(n-1)/q}-1,p)=1$, that is, $a^{(n-1)/q}\not\equiv 1 \pmod{p}$. It follows that $\operatorname{ord}_p(a)=ut$ for some $t \mid v$. But then $b\equiv a^t \pmod{p}$ has order u modulo p. But $\operatorname{ord}_p(b) \mid \phi(p)=p-1$, that is, $u \mid (p-1)$, that is, $p\equiv 1 \pmod{u}$.

(b) Conclude that if $u\geqslant\sqrt{n}$, then n is prime. **(5)**

Solution Suppose that n is composite. Take any prime divisor p of n with $p\leqslant\sqrt{n}$. By Part (a), $p\geqslant u+1\geqslant\sqrt{n}+1$, a contradiction. Therefore, n must be prime.

(c) Describe a situation when the criterion of Part (b) leads to an efficient algorithm for determining the primality of n. (**Hint:** Let all prime factors of u be *small*.) **(5)**

Solution In order to convert the above observations to an efficient algorithm, we need to clarify two issues.

(1) $n-1$ can be written as uv with u, v as above and with $u \geqslant \sqrt{n}$. We can keep on making trial divisions of $n-1$ by small primes $q_1 = 2$, $q_2 = 3$, $q_3 = 5$, ... until $n-1$ reduces to a value $v \leqslant \sqrt{n}$. If $n-1$ is not expressible in the above form, we terminate the procedure and report *failure* after a suitable number of small primes are tried.

(2) We need an element a satisfying the two conditions $a^{n-1} \equiv 1 \pmod{n}$ and $\gcd(a^{(n-1)/q} - 1, n) = 1$ for all $q|u$. If n is indeed prime, any primitive element modulo n satisfies these conditions, and there are at least $\phi(n-1)$ of them. This means that a suitable random base a is expected to be available within a few iterations.

4. Consider the subexponential expression

$$L_n(\omega, c) = \exp\left[c\,(\ln n)^{\omega}(\ln\ln n)^{1-\omega}\right]$$

for constants ω and c with $0 < \omega < 1$ and $c > 0$. Take $n \approx 2^{1024}$. Find the values of the expressions $n^{1/4}$, $L_n(1/2, 1)$ and $L_n(1/3, 2)$. What do these values tell about known integer-factoring algorithms? **(5)**

Solution We have the following approximate values:

$$\begin{aligned} n^{1/4} &= 2^{256} &&\approx 1.158 \times 10^{77}, \\ L_n(1/2,1) &\approx 2^{98.48} &&\approx 4.424 \times 10^{29}, \\ L_n(1/3,2) &\approx 2^{90.24} &&\approx 1.462 \times 10^{27}. \end{aligned}$$

These figures indicate that for factoring integers of bit size 1024, fully exponential methods (like Pollard's rho method) are very inefficient, the QSM is significantly faster than that, and the number-field sieve method is the fastest.

5. In the original QSM, we sieve around $\sqrt{n}$. Suppose we instead take $H = \left\lceil \sqrt{2n} \right\rceil$ and $J = H^2 - 2n$.

(a) Describe how we can modify the original QSM to work for these values of H and J. It suffices to describe how we get a relation in the modified QSM. There is no need to describe the sieving process or the linear-algebra phase, or to recommend optimal values for M (sieving limit) and t (size of the factor base). **(5)**

Solution For small values of c, consider the expression

$$(H + c)^2 \equiv H^2 + 2cH + c^2 \equiv J + 2n + 2cH + c^2 \equiv J + 2cH + c^2 \pmod{n}.$$

Call

$$T(c) = J + 2cH + c^2.$$

Since $H \approx \sqrt{2}\sqrt{n}$, and $J \leqslant 2\sqrt{2}\sqrt{n}$, it follows that $T(c) = \mathrm{O}(\sqrt{n})$ for small values of c. We choose a factor base $B = \{p_1, p_2, \ldots, p_t\}$ of t small primes. If the integer $T(c)$ factors completely over B, that is, if $T(c) = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_t^{\alpha_t}$, we get a relation

$$(H + c)^2 \equiv p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_t^{\alpha_t} \pmod{n}.$$

By varying c in the range $-M \leqslant c \leqslant M$, we collect s relations. The parameters M and t are so adjusted that one expects to get $s \gg t$ (like $s = 2t$). The resulting $t \times s$ system involving the exponents α_{ij} in the collected relations is solved to obtain a non-trivial congruence of the form $x^2 \equiv y^2 \pmod{n}$.

If a small prime p divides some $T(c)$, we have $(H + c)^2 \equiv 2n \pmod{p}$, that is, $2n$ is a quadratic residue modulo p. This implies that it suffices to include only those small primes in the factor base, modulo which $2n$ (not n) is a quadratic residue.

(b) Explain why the modified QSM is poorer than the original QSM. (**Hint:** Look at the approximate average value of $|T(c)|$.) **(5)**

Solution Since H and J are $\mathrm{O}(\sqrt{n}\,)$, and c is at most a subexponential expression in $\log n$, it follows that $|T(c)| \approx 2H|c|$ for almost all values of c. In other words, the absolute values of $T(c)$ are directly proportional to the value of H. For example, the average value of $|T(c)|$ approximately equals $2H \times (M/2) = MH$. In the original QSM, $H \approx \sqrt{n}$, whereas in the modified QSM, $H \approx \sqrt{2}\,\sqrt{n}$. This implies that the candidates tested for smoothness in the modified method are $\sqrt{2}$ times larger than those for the original QSM. For a fixed choice of M and t, the modified method is, therefore, expected to yield a smaller number of relations than the original method. Consequently, slightly larger values of M and/or t are needed by the modified method to yield sufficiently many relations, that is, the modified method is slightly more inefficient than the original method.

(c) Despite the objection in Part (b) about the modified QSM, we can exploit it to our advantage. Suppose that we run two sieves: one around $\sqrt{n}$ (the original QSM), and the other around $\sqrt{2n}$ (the modified QSM), each on a sieving interval of length half of that for the original QSM. Justify why this reduction in the length of the sieving interval is acceptable. Discuss what we gain by using the dual sieve. **(5)**

Solution Let $H = \lceil \sqrt{n} \rceil$ (original QSM) and $H' = \lceil \sqrt{2n} \rceil$ (modified QSM). Let M be the optimal sieving limit when the original QSM runs alone. In the case of the dual QSM, we run both the original QSM and the modified QSM with a sieving limit of $M/2$. In the original QSM, $2M + 1$ candidates (for $-M \leqslant c \leqslant M$) are tried for smoothness. In the dual QSM, there are two sieves each handling $M + 1$ candidates (for $-M/2 \leqslant c \leqslant M/2$). The total number of candidates for the dual QSM is, therefore, $2M+2$. In the original QSM, $2M+1$ candidates are expected to supply the requisite number of relations. So the dual QSM, too, is expected to supply nearly the same number of relations, provided that the candidates are not much larger in the dual QSM than in the original QSM. Indeed, we now show that the dual QSM actually reduces the absolute values of the candidates by a factor larger than 1.

The values of $|T(c)|$ for the original QSM are approximately proportional to H, whereas those for the modified QSM are roughly proportional to $H' \approx \sqrt{2}\,H$. In particular, the average value of $|T(c)|$ for the first sieve in this case is nearly $2H \times (M/4) = MH/2$ (the sieving interval is $M/2$ now). Moreover, the average value of $|T(c)|$ for the second sieve is about $2H' \times (M/4) \approx \sqrt{2}\,MH/2$. The average for both the sieves is, therefore, $(1 + \sqrt{2}\,)MH/4$. When the original QSM runs alone, this average is MH. Consequently, the smoothness candidates in the dual QSM are smaller than those for the original QSM by a factor of $4/(1 + \sqrt{2}\,) \approx 1.657$. As a result, the dual QSM is expected to supply more relations than the original QSM. Viewed from another angle, we can take slightly smaller values for M and/or t in the dual QSM than necessary for the original QSM, that is, the dual QSM is slightly more efficient than the original QSM.

The dual QSM does not consider the larger half of the $T(c)$ values (corresponding to $M/2 < |c| \leqslant M$) for smoothness tests. It instead runs another sieve. Although the smoothness candidates in the second sieve are about $\sqrt{2}$ times larger than the candidates in the original sieve, there is an overall reduction in the absolute values of $T(c)$ (averaged over the two sieves). This idea was first (apparently) proposed in my PhD thesis (in connection with the linear sieve method for computing discrete logarithms in prime fields—the LSM is a direct adaptation of the QSM).