Algebraic Cryptography
Exam 1

Choose 5 of the following 7 problems to complete.

Problem 1. Recall that in the RSA cryptosystem a user generates primes p, q and computes the product n. The user then generates an encryption exponent e relatively prime to $(p-1)$ and $(q-1)$. Then the user computes a decryption exponent d so that $de \equiv 1 \pmod{(p-1)(q-1)}$. The user then publishes the public key (n, e) and keeps the private key d a secret.

(a) Prove that the map $x \mapsto x^d \pmod{n}$ is the inverse of $x \mapsto x^e \pmod{n}$.

(b) Suppose Alice and Bob generate public keys (n, e) and (n', e') which share a prime p (i.e., $n = pq$ and $n' = pq'$), perhaps due to insufficient entropy (randomness) during key generation. Explain how an attacker, with knowledge only of the public keys can crack both Alice's and Bob's private keys.

Problem 2. Consider a commutative public key encryption decryption scheme (i.e., the encryption functions e and d commute, $e \circ d = d \circ e$). Let e_A, d_A be Alice's encryption/decryption functions, and let e_B, d_B be Bob's. Let h denote a hash function and m a message. The functions e_A, e_B, h are all public knowledge.

(a) Explain how Alice can use these functions to sign a message m in such a way that everyone (not just Bob) can read the message and everyone can verify that Alice wrote it.

(b) Explain how Alice can send a secret message to Bob that only he can read, but that Bob (and only Bob) can know and be sure that Alice wrote it.

Problem 3. Show that at least $\frac{n}{2}$ of the numbers $1, 2, \ldots, n$ have (binary) length equal to or greater than $\log_2 n - 1$. Then show that $n!$ has length at least equal to $\frac{n}{2}(\log_2 n - 2)$, and that for large n, this is greater than $Cn \ln n$ for some positive constant C.

Problem 4. Given a k-bit integer, you want to compute the highest power of this number that has l or fewer bits (we suppose $l \gg k$). Estimate (with big-O) the number of bit operations required to do this. Your answer should be a very simple expression in terms of k and/or l. Moreover, you should explicitly describe the algorithm you are using.

Problem 5. Suppose that you have a list of all primes having k or fewer bits. Using the Prime Number Theorem and big-O notation, estimate the number of bit operations needed to compute the sum of all these primes. You should explicitly describe the algorithm you are using. (*recall:* the number of bit operations required to add a k-bit number and an l-bit number is $\max\{k, l\}$.)

Problem 6. Let $\mathcal{P}_1$ be the decision problem

> **Input:** A polynomial $p(x)$ with integer coefficients.
> **Question:** Is there any interval of $\mathbb{R}$ on which $p(x)$ decreases?

Let $\mathcal{P}_2$ be the decision problem

> **Input:** A polynomial $p(x)$ with integer coefficients.
> **Question:** Is there any interval of $\mathbb{R}$ on which $p(x)$ is negative?

Show that $\mathcal{P}_1$ and $\mathcal{P}_2$ are equivalent ($\mathcal{P}_1$ reduces to $\mathcal{P}_2$ and $\mathcal{P}_2$ reduces to $\mathcal{P}_1$).

Problem 7. The Integer Factorization Search (IFS) Problem is the search problem

> **Input:** An integer $N > 1$
> **Output:** The statement "N is prime" or a nontrivial factor n of N.

The Integer Factorization Decision (IFD) Problem is the decision problem

> **Input:** An integer $N > 1$ and an integer k
> **Question:** Does N have a factor in the interval $[2, k]$?

Show that IFS reduces to IFD in polynomial time. That is, show there is a polynomial time algorithm (where the input size is the length of N) for IFS which makes at most polynomially many calls to an IFD-oracle.

Small bonus: Explain why IFD is in NP.