Introduction to Modern Cryptography, Assignment #3*

Benny Chor and Rani Hod

Published: 9/12/2009; revised: 15/12/2009; due: 24/12/2009, in Rani's mailbox (Schreiber, 2nd floor).

This assignment contains five "dry" problems and two "wet" problems. Efficient solutions are always sought, but a solution that works inefficiently is better than none. The answers to the the "wet" problem should be given as the output of a `Sage`, `maple` or `WolframAlpha` session.

1. **Weak Modifications of Encrypted Key Exchange**

   Recall the EKE method to perform an authenticated key exchange between Alice and Bob who share a secret password p taken from a dictionary D of size $|D| = 2^{30}$. Our prudent attacker Prudence has full control of the communication line, so she can read, change, silence or spoof any message; nevertheless, she does not want to be exposed so she cannot simply initiate communication with Alice 2^{30} times, trying all passwords from D.

   The EKE protocol works like this, where $Enc_k(x)$ is the symmetric 128-bit block encryption of the message x under the key k:[1]

select random a	$\xrightarrow{Enc_p(g^a)}$	
	$\xleftarrow{Enc_p(g^b)}$	select random exponent b
calculate $k = g^{ab}$		calculate $k = g^{ab}$
decipher z_b	$\xleftarrow{Enc_k(z_b)}$	select random block z_b beginning with the character 'b'
select random block z_a beginning with the character 'a'	$\xrightarrow{Enc_k(z_a, z_b)}$	decipher z_a, verify z_b
verify z_a	$\xleftarrow{Enc_k(z_a)}$	

   If verification fails, the protocol is immediately aborted. Pru can also halt all communication for a few minutes, fooling Alice and Bob to think that the other side has had a power outage, so Alice attempts to re-establish a connection to Bob and to renegotiate keys. Unless errors (connection or verification) happen a lot, Alice and Bob do not suspect.

   (a) Alice is lazy, so instead of sending $Enc_p(g^a)$ she actually sends g^a in the clear. Show how Pru can recover p without alarming Alice and Bob.

   (b) Did I say Alice? sorry, I meant Bob is lazy, so instead of sending $Enc_p\left(g^b\right)$ he actually sends g^b in the clear. Show how Pru can recover p without alarming Alice and Bob.

   (c) Ok, ok, Alice and Bob both encrypt their messages. But after they read Benny's presentation from lecture 5, they figured out an eavesdropper can find out whether k is a quadratic residue in $\mathbb{Z}_p^*$, so they decided to use odd a and b to make sure that the least significant bit of ab is one. Show them how stupid is this idea.

*Draft. Questions will not substantially change albeit might presented in a different wording.

[1] For simplicity, the verification steps were omitted in the recitation.

2. **Implementing RSA**

In this problem we will implement an instance of the RSA cryptosystem using `sage`. Start by choosing *at random* two prime numbers $p > 10^{82}$ and $q > 10^{77}$ such that $p-1$ has a prime factor greater than 10^{72} and $q-1$ has a prime factor greater than 10^{70}. Let $N = pq$. Pick at random e and calculate a matching d such that (e, d) are matching encryption and decryption RSA exponents.

(a) Print (with appropriate headings so we know what these numbers are) the numbers N, p, q, e and d, and also the complete factorizations of $p-1$ and of $q-1$. As a "scale for measuring lengths" print 10^{82} and 10^{72} as well so they are aligned with p and q respectively. Explain (in plain language, not in code) how p and q were found and especially how the random choices were made.

(b) Use the simple coding scheme presented in class (space=0, a=1, b=2, ..., z=26). Make up a short text, encode it (ASCII to numbers), encrypt it under your public key, then decrypt using the private key. Print the plaintext message, its encryption and the decryption.

(c) Team up with another working group. Get hold of their N and e. Encrypt your message from part (b) under these N and e. Send the result to that group and recieve a similarly encrypted message to you. Print the message you received, its decryption and its decoding as text. Print the name or names of the people you cooperated with, their public key, and the encrypted message you sent them.

(d) When decrypting a message, in addition to N and d you know also the factorization $N = pq$. This means that instead of calculating $m^d \bmod N$, it is possible to calculate $m^d \bmod p$ and $m^d \bmod q$ separately and then combine them to $m^d \bmod N$ using CRT. The benefit from this is that $\mathbb{Z}_p$ and $\mathbb{Z}_q$ are much smaller than $\mathbb{Z}_N$ so the two exponentiations are computed faster. Implement this improvement and measure how much time does it save you. Use larger p and q if you cannot see any difference.

In class Benny argued that the speed-up is approximately a factor of 4; was he essientially right?

Note: use the function `timeit` to find out how much time a calculation took.

3. **Pollard ρ Algorithm**

Recall that the Pollard ρ algorithm for factoring $N = p \cdot q$ utilizes a "random" function $F(z) = z^2 + c$ for some value of c. The fastest way to implement it is thus selecting $c = 0$, saving an addition.

Explain why this is a bad idea.

Hint 1: $c = -2$ is not better.
Hint 2: Try to analyze first the case where p and q are strong primes, that is, $p = 2p' + 1$, $q = 2q' + 1$ for prime p',q'.

4. **Rational Approximations of the $\log_2$ Function**

Assume we have a relation $2^n \approx p^m$; we then can get a rational approximation to $\log_2 p \approx \frac{n}{m}$. For instance, $2^{15} = 32768 \approx 32761 = 181^2$ and indeed $\log_2 181 = 7.4998\cdots \approx \frac{15}{2}$.

Similarly, from a relation $p_1^{n_1} p_2^{n_2} \approx p_3^{n_3} p_4^{n_4}$ we can deduce a linear equation in variables $\log_2 p_1$, $\log_2 p_2$, $\log_2 p_3$, $\log_2 p_4$. Collecting three such relations (along with the equation $\log_2 2 = 1$), we are able to solve the system.

(a) For a set B of primes, we call a number x B-smooth if all prime factors of x reside in B.

i. Let B be the set of primes up to 150 and let $M = 1000$. Collect enough numbers $x > M = 1000$ such that both x and $x+1$ are B-smooth.

ii. Generate and solve the resulting system of linear equations to get rational approximations of $\log_2 p$ for all p in B.

iii. Compare the approximation you get for $\log_2 99999999$ to the precise value. How off is it?

Hint: write a function that takes y and returns the `vector` $e \in \mathbb{N}^{|B|}$ such that $y = \prod p_i^{e_i}$. Then form a `matrix` m whose rows correspond to the linear equations and use `Sage`'s `m.solve_right` to get a solution. Make sure that you only pick independent rows, that is, that the matrix is square and has full rank — otherwise it might be inconsistent or there might be multiple solutions.

(b) Repeat the algorithm with $M = 10^6$. How is the accuracy effected? Estimate a lower bound for the accuracy for an arbitrary value of M.

5. **Low Exponent RSA**

In class, we discussed a problem that occurs for low exponent RSA (specifically we had $e = 3$) when the same plaintext x is encrypted with three different moduli ($x^3 \bmod N_i$ for $i = 1, 2, 3$). We stated, however, that there is no known problem when encrypting modulo a *single* N.

Suppose $N = pq$ is $2n$ bits long, p is of length $n+1$, q is of length $n-1$, and $n \geq 150$. Let $E(x) = x^3 \bmod N$, and let d denote the private decryption exponent. In this problem we will demonstrate that almost half the bits of d are easy to recover, without access to any secret information. Since in general $ed = 1 \bmod \varphi(N)$, we have in our case $3d = 1 \bmod \varphi(N)$. Therefore there is an integer A such that $3d - A\varphi(N) = 1$.

(a) Prove that $A = 2$.

(b) Show how to efficiently find an integer $\hat{d}$ satisfying $|d - \hat{d}| < \sqrt{N}$.

(c) Give a convincing argument (*not* a formal proof) why with high probability[2] d and $\hat{d}$ have the same $\frac{n}{2} - 5$ *most* significant bits.

6. **Square Roots and Factorization**

We are given a composite number N, which is n bits long, and we are told it is a product of two large primes $N = p \cdot q$. Recall that every square $x = z^2 \in \mathbb{Z}_{pq}^*$ has *four* square roots in $\mathbb{Z}_{pq}^*$.

Suppose we are now supplied with a blackbox *deterministic* algorithm $\mathcal{A}$ (we can feed it with several inputs and observe the outputs, but have no access to its internal working). On input $y \in \mathbb{Z}_{pq}^*$, $\mathcal{A}$ produces one of the following: If y is not a quadratic residue, then $\mathcal{A}$ outputs the text "`go fetch an Agama stellio for yourself`". If $y = x^2$ is a quadratic residue, $\mathcal{A}$ outputs *one* square root of y.

- Suppose on input y, $\mathcal{A}$ takes $t(n)$ steps. Show how to use $\mathcal{A}$ in order to factor N with high probability in $O(t(n))$ steps. Explain your analysis, and why randomization is essential in it.

7. **Hybrid Encryption and RSA-based Signatures**

Since RSA tends to be very slow compared to symmetric encryption, in practice a hybrid scheme is used. One possible incarnation of the hybrid system is the following:

- To encrypt a long message m employing Bob's public key (n, e), Alice selects a random 128-bit key k, computes $c_1 = AES_k(m)$ and $c_2 = k^e \bmod n$. She then sends the ciphertext $c = (c_1, c_2)$. Since k is much shorter than m, we only need one RSA operation instead of one per block.
- To decrypt a ciphertext $c = (c_1, c_2)$ with his private key (n, d), Bob computes $k = c_2^d \bmod n$ and then $m = AES_k^{-1}(c_1)$.

The hybrid system does actually provide improved efficiency (for long messages) without compromising security. Note that it is a public key cryptosystem, but not a deterministic one (which actually is advantageous). Can this hybrid system be employed for signing messages using the same paradigm we saw in class, proposed by Diffie and Hellman and implemented in "textbook RSA signature"?[3]

- Show how that paradigm is translated to the hybrid context described here.
- Prove that this scheme is totally insecure: after seeing Bob's signature on *any* message, Fritz the forger can create a valid signature of Bob on any other message.

[2] The probability here is over the choices of the random primes p and q.

[3] Reminder: Bob signs a message m by computing the (private) decryption of some hash function $h(m)$ and Alice can verify it by computing the (public) encryption of the signature.