CAS CS 538. Problem Set 5

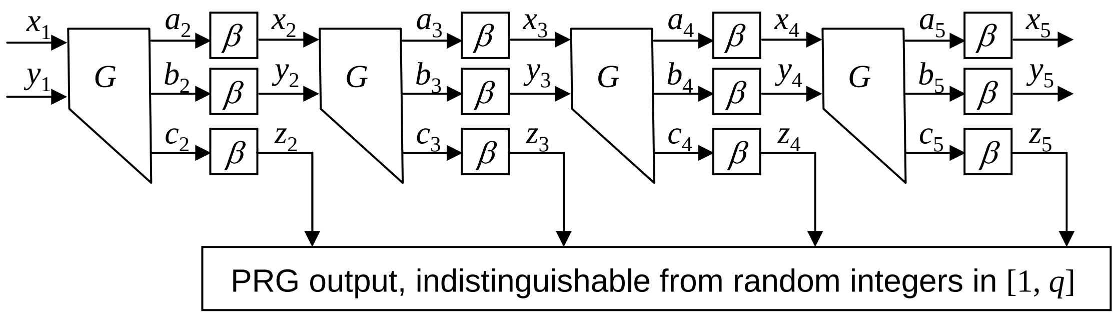
Due in class Tuesday, October 16, 2012, *before* the start of lecture

Problem 1. (25 points) In this problem, I want to show you an example of very real challenges that come from implementing cryptography in real life. In this example, the adversary can do more than what we allowed it to do in mathematical model: it can tamper with the computation. Theoretical cryptography has only recently started modeling and addressing tampering adversaries, and interesting new schemes are coming out to address them.

Suppose you have a device (smart card, computer, etc.) that is performing an RSA secret-key operation (computing $m = c^d \bmod n$ for some c) using CRT: separately computing $m_p = c^d \bmod p$ and $m_q = c^d \bmod q$ and then combining. Suppose you can hit the device with just enough radiation or power-glitch it to cause exactly one of the two modular exponentiations to compute an incorrect value. That is, you get $m'_p \neq m_p$ but m_q is correct. This causes the output to be some $m' \neq m$. Show how to factor n given c and m' (and, of course, the RSA public key (n, e)). This is an actual attack that can be carried out on certain smart cards. (Hint: it may be easier to first figure out how to factor n given m, m', and the public key (n, e).)

Problem 2. (25 points) Lamport's signatures for a message space of size 2^ℓ have 2ℓ values in the secret key (and, therefore, also in the public key). A signature consists of a particular message-dependent subset of those values. For simplicity, we will focus on $\ell = 3$: thus, the secret key consists of 6 values and the message space has size 8. Show how to generalize Lamport's scheme to allow for signatures on the message space of size 20 (i.e., 20 different messages, not 20 bits of message) while keeping the key size at 6 values.

Problem 3. (25 points) One reason people consider the Decisional Diffie-Hellman assumption to be very strong is that it makes it too easy to build a PRG: in a sense, it already assumes a PRG. We will build a PRG out of DDH in this problem, as described in the following picture.



For simplicity, our PRG will not output bit strings that are indistinguishable from random, but rather sequences of integers between 1 and q that are indistinguishable from random.

Let q be a Sophie Germain prime, $p = 2q + 1$, and g be a generator of QR_p. Assume that p and g are public (or, alternatively, are part of the seed to our PRG). Assume DDH hold in QR_p. Let G be the function take two integers x_1, y_1 from $[1, q]$ and outputs three elements of QR_p,

$a_2 = g^{x_1}, b_2 = g^{y_1}, c_2 = g^{x_1 y_1}$. DDH says that a_2, b_2, c_2 are indistinguishable in polynomial-time from random elements of QR_p.

Now that you have three elements of QR_p, you'd like to map them back to three elements of $[1, q]$. The only work you need to do in this problem is this: show that the following function β is a bijection from QR_p to $[1, q]$ that is efficiently computable both ways: $\beta(a) = \{a$ if $a \leq q$, and $p - a$ otherwise$\}$. Hint: use the fact that $p \equiv 3 \pmod 4$ (but first show why that's the case).

Now the final PRG is built as follows (you don't need to prove anything for this paragraph; the proof would be a hybrid argument): start with two random elements from $[1, q]$, apply G, and apply β to each of the three elements of QR_p that result from G. Now you have three elements from $[1, q]$ that look random. Output one of them, and use the other two as an input to G again. Iterate as many times as needed.

Problem 4. (25 points) In class we considered only passive adversaries for public-key encryption: our adversaries have been limited to observing ciphertexts. A stronger adversary is one endowed with the power to obtain decryptions of some ciphertexts; we considered this kind of adverary for symmetric encryption. Notice that this stronger adversary is quite realistic: for example, when someone sends you email, you often include whatever was sent in your reply. So if you use encrypted email, the adversary could send you a ciphertext and wait for the reply to find out what the plaintext was. This is known as a "chosen ciphertext attack."

There are two types of chosen ciphertext attack. In the first one, the adversary gets to ask for decryptions of ciphertexts *before* she gets the challenge ciphertext c (which is an encryption of m_0 or m_1). In the second one, she also gets to ask *after* seeing c (but, of course, she doesn't get to ask for a decryption of c — that would be impossible to protect against; however, she may ask for ciphertexts related to c if she wants). There are denoted CCA1 and CCA2, respectively. (CCA1 is also sometimes referred to as "lunchtime" attack, because, presumably, adversary just gets to you use your computer for a while when you are out to lunch. CCA2 is sometimes called a "midnight" attack, for reasons that I don't comprehend.) While CCA1 is hard to protect against and CCA2 is even harder, there exist cryptosystems secure against CCA1 and CCA2 under reasonable assumptions.

However, all the cryptosystems we studied in class so far fail under CCA2. In this problem, you will show one specific attack. Namely, show a CCA2 attack on ElGamal encryption (hint: modify the challenge ciphertext c and ask for its decryption; then "undo" the modification).