CS-7880 Graduate Cryptography December 1, 2015

Problem Set 3

Lecturer: Daniel Wichs *Due: Dec. 10, 2015*

Problem 1 (Public Key Encryption – Decryption Query) 20 pts

The security definition of public-key encryption that we gave in class gives the adversary the public key which allows him to encrypt arbitrary messages himself. However, it doesn't consider that an adversary might be able to see how ciphertexts are decrypted. In this problem, you're to show that in general this can make a cryptosystem completely insecure.

A. Show that, if there exists any secure public key encryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ according to the definition we gave in class then you can modify it to get an encryption scheme $\mathcal{E}' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ such that:

- $\mathcal{E}'$ is a secure encryption scheme according to the definition we gave in class.
- $\mathcal{E}'$ has the property that, if the attacker can query the decryption function $\mathsf{Dec}'(sk, \cdot)$ even on a single ciphertext c of his choosing and see the output $m = \mathsf{Dec}(sk, c)$ then the attacker can completely recover the secret key sk.

This is a very undesirable property - if the attacker can learn a single decrypted value (for a ciphertext of his choosing) he can completely break security of the scheme. We will see how to define stronger notions of security that prevent this later on in the class.

B. You solution in part A might have been a "contrived" scheme which is not very "natural". But there are natural schemes that are completely insecure if an adversary can see decryptions of chosen messages – for example, schemes based on the Rabin trapdoor permutation. Let $N = pq$ be a product of two primes and let $f : QR_N \to QR_N$ be the Rabin trapdoor permutation defined by $f(x) = x^2 \mod N$. We know this permutation is easily invertible given p, q. Show that if an adversary can query $f^{-1}(y)$ for a single value y of its choosing than it can efficiently factor N with non-negligible probability.

Problem 2 (Hedging You Bets) 5 pts

Assume you have two public-key encryption schemes $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ and $\mathcal{E}' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$. You believe that at least one of them has to be secure but not necessarily both. Construct an encryption scheme $\mathcal{E}^* = (\mathsf{KeyGen}^*, \mathsf{Enc}^*, \mathsf{Dec}^*)$ which is secure as long as at least one of $\mathcal{E}$ or $\mathcal{E}'$ is secure.

Problem 3 (Commitments) 20 pts

A commitment scheme allows Alice to "commit" herself to some message m by giving Bob some value $c = \mathsf{Commit}(m, r)$ generated using randomness r. Bob should not learn anything about m given the commitment c. Later she can "open" the commitment by giving (m, r) to Bob to convince him that m was the value she committed herself to. We will assume that $m \in \{0, 1\}$ is just a single bit – we can always extend this to a longer message by committing one bit at a time.

Formally, a commitment is a function $\mathsf{Commit} : \{0, 1\} \times \{0, 1\}^* \to \{0, 1\}^*$ that should satisfy the following properties.

- Hiding: The commitments to 0 and 1 are computationally indistinguishable $\mathsf{Commit}(0, U_n) \approx \mathsf{Commit}(1, U_n)$ where U_n denotes the uniform distribution over $\{0, 1\}^n$.
- Binding: For all PPT adversaries A, we have

$$\Pr[\mathsf{Commit}(0, r) = \mathsf{Commit}(1, r') : (r, r') \leftarrow A(1^n)] = \mathsf{negl}(n).$$

A. Show that commitments imply the existence of one-way functions.

B. You will now show how to construct a generalized notion of such commitments, which we'll call seeded commitments, from one-way functions. A seeded commitment also contains a seed s and we define the commitment function as $\mathsf{Commit}_s(m, r)$ which takes the seed s as an input. We think of Bob as generating the seed s and therefore we want hiding to hold even if s is chosen maliciously. On the other hand, we want binding to hold when s is chosen randomly. Give a formal definition of the hiding and binding properties for seeded commitments to capture this intuitive description.

Consider the following scheme: Let G be a PRG with $2n$-bit stretch, so that for $|r| = n$, $|G(r)| = 3n$. Let $s \leftarrow \{0, 1\}^{3n}$ be a random string of length $3n$. Define $\mathsf{Commit}_s(m, r)$ to be $G(r)$ if $m = 0$ and $G(r) \oplus s$ if $m = 1$. Show that this scheme satisfies the definition of seeded commitments, and that the binding property holds even if the adversary is computationally unbounded.

C. Show that the above scheme could potentially be insecure if we defined s to be some fixed string, say all 1s, rather than choosing it randomly. To do so, show that if PRGs exist then there exists a PRG G for which such scheme would be insecure.

D. In the scheme from part B, the hiding property holds when the adversary is computationally bounded but binding holds even if the adversary is computationally unbounded. We could ask whether the reverse is also possible.

Consider the following commitment scheme based on the discrete-logarithm assumption. The seed s consists of $s = (\mathbb{G}, q, g, h)$ where $(\mathbb{G}, q, g) \leftarrow \mathsf{GroupGen}(1^n)$ is a description of a cyclic group $\mathbb{G}$ of prime order q with generator g, and $h \leftarrow \mathbb{G}$ is a random group element. We define $\mathsf{Commit}_s(m; r) = g^m h^r$ where $m \in \{0, 1\}$, $r \in \mathbb{Z}_q$. (We're changing the syntax a little so that the randomness is uniform over $\mathbb{Z}_q$ rather than $\{0, 1\}^n$. We could also allow m to come from all of $\mathbb{Z}_q$ rather than just $\{0, 1\}$ but let's stick with 1-bit messages to keep the syntax and the definitions consistent).

Show that the above scheme satisfies perfect hiding: for any s (not necessarily random) the distributions $\mathsf{Commit}_s(0, r) \equiv \mathsf{Commit}_s(1, r)$ are identical over the choice of a random r. Show that, under the discrete logarithm assumption, the scheme is binding when s is chosen randomly as specified above.