CS/ECE407 Cryptography – Homework 5 Solutions
Public Key Cryptography

Due: Wednesday, May 6, 11:59PM CT

Remember, you are free to collaborate with up to one classmate. See the course webpage for more details. You are expected to write out and submit your own solutions! Your collaboration is for discussing problems at a high level, not plagiarizing answers.

Your homework should be typed or carefully handwritten. We provide a LaTeX template for this document, if you would like to use it as a starting point. **If we cannot read your handwritten answers, they will not receive credit.**

Your typed solutions should be submitted through gradescope (see course webpage). Your hand-written solutions should be scanned and turned in through gradescope.

Problem 1 (Certificates)**.** Recall that a certificate is a document signed by some 'certificate authority' (CA) that associates an identity with a public key. For instance, let sk_C be the secret key of the CA. The following is a certificate associating Bob with some key pk_B:

$$Cert_{C \to B} = Sign(sk_C, \text{“Bob's key is } pk_B\text{”})$$

This problem involves *certificate revocation*, where, for example, a CA might maintain a list of certificates that should no longer be trusted. In the following, you may assume that the CA wishes to revoke any certificate for which the secret key has been compromised.

1. **(2 points)** Suppose Bob's secret key is leaked in a data breach, and so now he would like the CA to revoke this certificate, so that others cannot forge messages on his behalf. Bob sends the following message to the CA:

   $$m = \text{“please revoke } Cert_{C \to B}\text{”}$$

   The CA calls Bob on the phone to verify Bob's identity using non-cryptographic mechanisms. Why does the CA need to verify Bob's identity before revoking his certificate?

2. **(2 points)** Now, in addition to sending m, suppose that Bob also signs m and sends the following signature to the CA:

   $$\sigma = Sign(sk_B, m)$$

   Does the CA still need to call and verify Bob's identity before revoking his certificate? Why/why not?

Solution 1.

1. If the CA does not verify Bob's identity, then an adversary can revoke Bob's certificate at any time. This creates a kind of denial-of-service attack, where an adversary can remove certificates without knowing the secret key.

2. No, the CA does not need to wait before revoking Bob's certificate. (Though, for practical reasons, it is probably still a good idea for the CA to let Bob know that the revocation has happened.) The reasoning here is that there are are only two possibilities:

   - Bob requested the revocation. In this case, revoking the certificate is the intended behavior.
   - An adversary requested the revocation. But if the adversary successfully created a signature under Bob's signing key, then that adversary *must know* sk_B. That is, Bob's key has been compromised, and hence it should be revoked!

Problem 2 (RSA)**.** Suppose Enc', Dec' is a symmetric-key encryption scheme with CPA security where keys are uniformly sampled from $\{0,1\}^\lambda$.

> Clarification: You may assume that Enc' satisfies the following (weaker) notion of security. The security game is identical to the CPA security game except instead of sampling a single key k and using it for all Enc_L or Enc_R queries, every query is answered using a freshly sampled key k. The security requirement is the same: No PPT adversary can distinguish Enc_L and Enc_R.

Recall that the RSA key generation algorithm outputs a triple (N, e, d) such that:

$$e \cdot d \equiv 1 \mod \phi(N)$$

Let $H : \mathbb{Z}_N^* \to \{0,1\}^\lambda$ be a random oracle, and consider the following public-key encryption algorithm that uses RSA keys:

$$\boxed{\begin{array}{l} Enc((N,e),m): \\ \quad r \leftarrow \mathbb{Z}_N^* \\ \quad k = H(r) \\ \quad \mathsf{return}\ (r^e \bmod N, Enc'(k,m)) \end{array}}$$

1. **(2 points)** Specify a corresponding decryption algorithm:

$$\boxed{\begin{array}{l} Dec((N,d),c): \\ \quad ... \end{array}}$$

2. **(2 points)** Prove that your decryption algorithm is correct.

3. **(2 points)** Argue that your public key scheme is CPA secure, under the RSA assumption. Namely, argue why it is that an adversary that can win the CPA game must be able to break either the CPA security of the symmetric-key scheme, or the RSA assumption.

Solution 2.

1.

$$\begin{array}{l} Dec((N,d),(c_0,c_1)): \\ \quad r = c_0^d \bmod N \\ \quad k = H(r) \\ \quad m = Dec'(k,c_1) \\ \quad \mathsf{return}\ m \end{array}$$

2. We must prove that for all m, the following holds with probability 1:

$$\Pr\left[\ Dec((N,d),c) = m \ \middle|\ \begin{array}{l} (N,e,d) \leftarrow RSAGen(1^\lambda) \\ c \leftarrow Enc((N,e),m) \end{array}\ \right]$$

   Inlining the definition of Enc, Dec, we get the following:

$$\Pr\left[\ m' = m \ \middle|\ \begin{array}{l} (N,e,d) \leftarrow RSAGen(1^\lambda) \\ r \leftarrow \mathbb{Z}_N^* \\ k = H(r) \\ c_0 = r^e \bmod N \\ c_1 \leftarrow Enc'(k,m) \\ r' = c_0^d \bmod N \\ k' = H(r') \\ m' = Dec'(k',c_1) \end{array}\ \right]$$

Now, we know that $r' = c_0^d \bmod N = r^{de} \bmod N$. But we also know that $de \bmod \phi(N) = 1$ and hence $r^{de} = r \bmod N$. Thus, $k' = k$. Hence, we can simplify the above probability:

$$\Pr\left[\begin{array}{l|l} & (N,e,d) \leftarrow RSAGen(1^\lambda) \\ m' = m & r \leftarrow \mathbb{Z}_N^* \\ & k = H(r) \\ & c_1 \leftarrow Enc'(k, m) \\ & m' = Dec'(k, c_1) \end{array}\right]$$

But this is just the probability that the underlying symmetric key scheme is correct, which is 1. Hence, the scheme is correct.

3. Consider a ciphertext of form $(r^e, Enc'(H(r), m_0))$. We roughly wish to argue that this ciphertext hides m_0.

   Now, since r is uniform over $\mathbb{Z}_N^*$, we can use the RSA assumption to argue that no adversary can recover r from r^e. Since H is a random oracle, the value $H(r)$ is independent of the adversary's view, unless the adversary successfully queries H at r. But to issue such a query, the adversary must produce r with noticeable probability, where the only relevant information in its view is r^e. Thus, to issue query $H(r)$, the adversary must break RSA.

   Therefore, under the RSA assumption, we may replace $H(r)$ by a fresh, uniform key k: That is, in a hybrid experiment we can consider encryptions of the form $(r^e, Enc'(k, m_0))$. But the security of our symmetric-key scheme is precisely strong enough to show that we can replace $Enc'(k, m_0)$ by $Enc'(k, m_1)$ (when k is uniform). In other words, an adversary that distinguishes these two must break our symmetric-key scheme.

   To complete the proof, one can rewrite $(r^e, Enc'(k, m_1))$ to $(r^e, Enc'(H(r), m_1))$ using the same reasoning about RSA as argued above.

Problem 3 (CCA Security for Public-Key Schemes)**.** In class we studied public-key schemes with CPA security. It is useful to also define *CCA security* for public-key encryption schemes. Recall that in the symmetric-key setting, CCA security is related to *malleability* – namely the ability for the adversary to induce a change in a message by changing the encryption of that message. The same is true for public-key schemes, and the definition of CCA security is similar:

Definition 1 (Public Key Encryption Security against Chosen Ciphertext Attacks)**.** Let $\Pi = (KeyGen, Enc, Dec)$ denote a public-key encryption scheme. We define two helper procedures:

$$Enc_0(pk, m_0, m_1) = \{\ c_0 \mid c_0 \leftarrow Enc(pk, m_0)\ \}$$
$$Enc_1(pk, m_0, m_1) = \{\ c_1 \mid c_1 \leftarrow Enc(pk, m_1)\ \}$$

We say Π achieves CCA security if the following ensembles are indistinguishable to any polytime adversary issuing *legal queries* (discussed next):

$$\{\ (pk, Dec(sk, \cdot), Enc_0(pk, \cdot, \cdot)) \mid (sk, pk) \leftarrow KeyGen(1^\lambda)\ \} \approx$$
$$\{\ (pk, Dec(sk, \cdot), Enc_1(pk, \cdot, \cdot)) \mid (sk, pk) \leftarrow KeyGen(1^\lambda)\ \}$$

That is, the adversary is given the public key pk, as well as oracle access to decryption, and oracle access either to Enc_0 or Enc_1. The *legal query* restriction states that the adversary may not pass any output obtained from its encryption oracle as input to its decryption oracle.

1. **(3 points)** Recall the ElGamal encryption scheme, which we discussed was CPA secure under the DDH assumption. Show that ElGamal is *not* CCA secure.

Solution 3.

1. Suppose ElGamal is defined over group $\mathbb{G}$ with order q and generator g. Let sk denote a secret key, and let $h = g^{sk}$. Recall that a ciphertext corresponding to message $m \in \mathbb{G}$ is as follows:

$$c = (c_0, c_1) = (g^r, h^r \cdot m) \qquad \text{where } r \leftarrow \mathbb{Z}_q$$

Recall that decryption proceeds by computing:

$$\frac{c_1}{c_0^{sk}} = \frac{h^r \cdot m}{(g^r)^{sk}} = \frac{g^{sk \cdot r} \cdot m}{g^{sk \cdot r}} = m$$

Now, we can construct an adversary that breaks CCA security as follows:

- Pass two messages $1 \in \mathbb{G}$ and $g \in \mathbb{G}$ to the encryption oracle. Let the resulting ciphertext be $c = (c_0, c_1)$.
- The adversary computes a new ciphertext:

$$c' = (c_0 \cdot c_0, c_1 \cdot c_1) = (c_0^2, c_1^2)$$

- The adversary passes c' to the decryption oracle.
- If the decryption oracle returns $1 \in \mathbb{G}$, the adversary outputs 0; else it outputs 1.

The above attack works with overwhelming probability. To see this, let c be an encryption of some message m. Our claim is that c' decrypts to m^2:

$$\frac{c_1^2}{(c_0^2)^{sk}} = \frac{h^{2r} \cdot m^2}{(g^{2r})^{sk}} = \frac{g^{sk \cdot 2r} \cdot m^2}{g^{sk \cdot 2r}} = m^2$$

Moreover, c' is distinct from c (unless $r = 0$ and $m = 1$; the former occurs with negligible probability). Thus, this decryption query is legal. Since $1 \neq g$, when given Enc_0, the adversary's decryption query returns $1 \in \mathbb{G}$ and they output 0; When given Enc_1, the adversary's decryption query returns $g^2 \neq 1$ and they output 1. Hence, ElGamal is not CCA secure.