CSE 426: Cryptography
Prof. Stefano Tessaro

University of Washington
Autumn 2023

Homework 6

Posted: Monday, November 20, 2023 – 11:59pm
Due: Friday, December 1, 2023 – 11:59pm

Task 1 – RSA Modulus Generation (10 points)

Dr. Crypto proposes the following method to generate a $2k$-bit RSA modulus $N = P \times Q$. (E.g., we may have $k = 2048$, and hence $2k = 4096$.)

1. We pick a random k-bit integer R.
2. We compute $R-1$, $R-2$, $\ldots$, until we find a prime $R-i$, and set $P = R-i$.
3. We compute $R+1$, $R+2$, $\ldots$, until we find a prime $R+j$, and set $Q = R+j$.

The final output is $N = P \times Q$. Because primes are dense, this algorithm will only take a polynomial number of steps (in k) to find a suitable P and Q, and moreover, testing whether a number is prime can also be done efficiently.

a) [6 points] Explain why this generation method, while correct (i.e., it does generate a $2k$-bit modulus N), leads to insecure public keys when used for RSA-based cryptography.

b) [4 points] Assume the following RSA modulus is generated using the above method for some choice of k we do not know explicitly. (N is one single integer, with 124 decimal digits, but it is split over two lines for space reasons!)

$$N = \texttt{12336261539757652568320691057196254494530050076556470009232333}\\ \texttt{67120767290238588667397052161653352801437540471197470570083267}.$$

Write a small Python program to find P and Q. Include, P, Q, and the code in the solution PDF.

Hint: Python handles large integers well, and thus may be helpful. You should also beware of any floating point functions and operations, as their outputs would have very bad precisions.

Task 2 – ElGamal and DDH (15 points)

We define a public-key encryption scheme $\Pi = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathcal{M} = \{0, 1\}$ that relies on a cyclic group $\mathbb{G} = \langle g \rangle$ with generator g and prime order p. The key generation algorithm is identical to that of El-Gamal, i.e.,

- Kg samples $x \xleftarrow{\$} \mathbb{Z}_p$, and outputs $\mathrm{PK} = h = g^x$ and $\mathrm{SK} = x$.

The encryption algorithm $\mathsf{Enc}(\mathrm{PK}, b)$ encrypts a bit $b \in \{0, 1\}$ as follows:

- If $b = 0$, then choose a random $y \xleftarrow{\$} \mathbb{Z}_p$, and compute $C_1 = g^y$ and $C_2 = \mathrm{PK}^y$. Output the ciphertext $C = (C_1, C_2)$.

- If $b = 1$, then choose independently random $y \xleftarrow{\$} \mathbb{Z}_p$ and $z \xleftarrow{\$} \mathbb{Z}_p$, and compute $C_1 = g^y$ and $C_2 = g^z$. Output the ciphertext $C = (C_1, C_2)$.

a) [5 points] Describe the decryption algorithm $\mathsf{Dec}(\mathrm{SK}, C = (C_1, C_2))$. Show that the correctness requirement of the scheme may sometimes not hold, but that the output of Dec is incorrect only with very small probability.

We now want to show that Π is IND-CPA secure under the DDH assumption. As we discussed in class, it suffices to prove that Π satisfies the notion of *one-time* IND-CPA security which asks for indistinguishability of the oracles $\mathsf{1\text{-}LR}_0[\Pi]$ and $\mathsf{1\text{-}LR}_1[\Pi]$, defined in the following figure, that only allow a *single* Encrypt query. In particular, Π is one-time IND-CPA secure if

$$\mathrm{Adv}_{\Pi}^{\text{1-ind-cpa}}(D) = \left|\Pr[D^{\mathsf{1\text{-}LR}_0[\Pi]} \Rightarrow 1] - \Pr[D^{\mathsf{1\text{-}LR}_1[\Pi]} \Rightarrow 1]\right|$$

is negligible for all polynomial-time D.

```
oracle 1-LR_b[Π]:        // b ∈ {0,1}

private procedure Init():
(PK, SK) ←$ Kg(); queried ← 0
return PK

public procedure Encrypt(M^0, M^1):
if |M^0| ≠ |M^1| or queried = 1 return ⊥
C ←$ Enc(PK, M^b); queried ← 1
return C
```

```
oracle DDH_b[G, g]:        // b ∈ {0,1}

private procedure Init():
x, y, z ←$ Z_p
if b = 0 then
    return (g^x, g^y, g^{xy})
else
    return (g^x, g^y, g^z)
```

On the right, we also recall the oracles $\mathsf{DDH}_0[\mathbb{G}, g]$ and $\mathsf{DDH}_1[\mathbb{G}, g]$ used to define the DDH assumptions in $\mathbb{G}$ with respect to g, which holds if

$$\mathrm{Adv}_{\mathbb{G},g}^{\mathrm{ddh}}(D) = \left|\Pr[D^{\mathsf{DDH}_0[\mathbb{G},g]} \Rightarrow 1] - \Pr[D^{\mathsf{DDH}_1[\mathbb{G},g]} \Rightarrow 1]\right|$$

is negligible for all polynomial-time D.

b) [8 points] Specify the pseudocode description of an oracle O such that $\mathsf{O}^{\mathsf{DDH}_b[\mathbb{G},g]} \equiv \mathsf{1\text{-}LR}_b[\Pi]$ for both $b = 0$ and $b = 1$. Here, we allow O's Init to take an input (X, Y, Z), which is the output returned by the Init procedure of the given oracle $\mathsf{DDH}_b[\mathbb{G}, g]$.

c) [2 points] Use **b)** to conclude that Π is one-time IND-CPA secure if the DDH assumption holds in $\mathbb{G}$ with respect to g.

Hint: If you use the notation $D' = D^{\mathsf{O}}$, in this case the input of D' is first passed through the Init procedure of O.

Task 3 – Chosen-Ciphertext Security (10 points)

Consider the ElGamal scheme over a cyclic group $\mathbb{G} = \langle g \rangle$ with generator g and order p. Recall that the secret key is an element $\mathrm{SK} \in \mathbb{Z}_p$, the public key is $\mathrm{PK} = g^{\mathrm{SK}}$, and the encryption of a plaintext $M \in \mathbb{G}$ produces the ciphertext $C = (g^r, M \star \mathrm{PK}^r)$, where "$\star$" denotes the group operation in $\mathbb{G}$, which we assume to be efficient.

a) [2 points] Given an ElGamal ciphertext C encrypting an (unknown) message $M \in \mathbb{G}$, how can you (efficiently) produce a ciphertext C' encrypting the message $M \star \delta$, for a known $\delta \in \mathbb{G}$?

b) [2 points] Argue that if $M_1, M_2 \in \mathbb{G}$ are distinct, i.e., $M_1 \neq M_2$, then for any $\delta \in \mathbb{G}$, the group elements $M_1 \star \delta$ and $M_2 \star \delta$ are also distinct.

c) [6 points] Use **a)** and **b)** to show that ElGamal encryption is *not* IND-CCA secure. In particular, give a distinguisher D and show that $\mathrm{Adv}^{\text{ind-cca}}_{\Pi}(D) = 1$.

Task 4 – AES-Based Signatures (15 points)

Alice proposes the following signature scheme $\Sigma = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Ver})$, based on the security of AES (with 128-bit keys) to sign n-bit messages.

- Kg chooses $2n$ 128-bit random strings $K_{1,0}, K_{1,1}, \ldots, K_{n,0}, K_{n,1} \xleftarrow{\$} \{0,1\}^{128}$, and outputs
$$\mathrm{SK} = (K_{1,0}, K_{1,1}, \ldots, K_{n,0}, K_{n,1}) \ , \ \ \mathrm{VK} = (C_{1,0}, C_{1,1}, \ldots, C_{n,0}, C_{n,1}) \ ,$$
where $C_{i,b} = \mathrm{AES}(K_{i,b}, 0^{128})$ for all $i = 1, \ldots, n$ and $b = 0, 1$.

- $\mathsf{Sign}(\mathrm{SK}, M)$, given $M \in \{0,1\}^n$, outputs $\sigma = (\sigma_1, \ldots, \sigma_n)$, where $\sigma_i = K_{i,M_i}$ with M_i being the i^{th} bit of M.

- $\mathsf{Ver}(\mathrm{VK}, M, \sigma)$ outputs 1 if and only if $\mathrm{AES}(\sigma_i, 0^{128}) = C_{i,M_i}$ for all $i = 1, \ldots, n$.

For **a)** and **b)**, you can assume that for any "efficient" adversary, given $\mathrm{AES}(K, 0^{128}) = C$ for a uniform 128-bit key K, it is hard to find K' such that $\mathrm{AES}(K', 0^{128}) = C$, except with very small probability. (We do not make "efficient" and "small" formal here, and you can keep this informal throughout **a)** and **b)**.)

a) [4 points] Argue that the above scheme satisfies key-only unforgeability, i.e., given only VK output by Kg, it is computationally hard for an adversary to come up with a message M^* and a valid signature σ^* such that $\mathsf{Ver}(\mathrm{VK}, M^*, \sigma^*)$ outputs 1.

You do not have to write a reduction.

b) [4 points] Argue that what you have shown in **a)** remains true even if the adversary sees a valid signature σ for a message M of their choice, and attempts to come up with a second signature σ^* for a message $M^* \neq M$ as the forgery.

You do not have to write a reduction.

Hint. Why can you not easily compute σ^* from σ for any $M^* \neq M$?

c) [7 points] Let $n \geq 2$. Show that Σ is, in general, not UF-CMA secure, by giving a concrete adversary A such that $\mathrm{Adv}^{\text{uf-cma}}_{\Sigma}(A) = 1$. In view of **b)**, the adversary A should obtain at least two signatures before producing a forgery.