Isaac Kim
CS 487
Professor Baldimtsi
Apr 29, 2023

Homework 6

1.1 Let x be the secret key, c be our ciphertext and c_1, c_2 be the two halves of our ciphertext. Decryption works as follows: compute c_1^x and compare with c_2. Then, there are two cases:
Case 1: $c_2 = h^y$, $m =$ head

In this case, $c_1^x = (g^y)^x = g^{xy} = (g^x)^y = h^y = c_2$. Then, our message is "head".

Case 2: $c_2 = g^z$, $m =$ tail

In this case, $c_1^x = (g^y)^x = g^{xy} \neq g^z = c_2$. Then, our message is "tail". It is important to note that in this scenario, there is the possibility that $z = xy$ and we can decrypt incorrectly, but with a sufficiently large q, the probability of this happening is negligible.

1.2 Contrapositive: If the above scheme is *not* CPA secure, then DDH is "easy" in G. (DDH problem: Given g, h_1, h_2, distinguish $\text{DH}_g(h_1, h_2)$ from a uniform element of G)

This implies that there exists an adversary A that breaks the CPA security of the scheme such that Pr[A wins] $\geq \frac{1}{2} + p(n)$ where $p(n)$ is non-negligible.

Let B be our adversary for DDH. B will receive a DDH instance (h_1, h_2, w) as input from the challenger, where $h_1 = g^x, h_2 = g^y, w = g^{xy}$ or a random string. B sets $pk = h_1$ and sends pk to A. A sends its challenge $m_0 = head, m_1 = tail$. B flips a bit $b \in \{0, 1\}$ and computes encryption of m_b such that $Enc_{pk}(m_b) = c*$ according to the encryption scheme in 1.1. Then, B sends $c*$ to A. A will output a bit b'. If $b' = b$, B will output that w is a "DDH tuple", else output that w is random.

Analysis:

Case 1: $w = g^{xy}$, $pk = x$, $c_1 = g^y$, $c_2 = h^y = (g^x)^y = g^{xy}$

Here, Pr[B wins] = Pr[A wins]

Case 2: $w =$ random string

Here, Pr[A wins] $= \frac{1}{2} =$ Pr[B wins]

2.1 Let A be our adversary. Note, for some message m, $m' = 00000000||r||00000000||m$, $|m'| = ||N||$. A gets access to some ciphertext $c = [m^e \bmod \text{N}]$. A then calculates $c_\beta = [\alpha^e \cdot c \bmod N]$ for some $\alpha \in \mathbb{Z}$. Then, A sends c_β to the decryption oracle. $Dec(c_\beta)$ will either return an error or give back a proper decyption $m'_\beta = [c_\beta^d \bmod \text{N}]$. In the case that A gets an error, it can just keep trying different $\alpha \in \mathbb{Z}$ until it gets back a valid m'_β. Then, since $m'_\beta = \alpha \cdot m'$, A can compute m'_β / α and extract m from m'.

2.2 It is easier to construct a chosen-ciphertext attack on this scheme than on PKCS#1 v1.5 because the padding is not of random length. In our scheme, the padding is of a set length, and r is also short (8 random bits).

3.1 Existential unforgability: An attacker should be unable to forge valid signature on *any* message not signed by the sender.

Let A be our attacker that is given the public key. A can interact with the oracle $\text{Sign}_{sk}()$. By Kerckhoff's principle, A knows how the signing works. Then, in order to forge message $M = m_1||m_2||...||m_n$, A can interact with the sign oracle to sign message $M_\alpha = m_n||m_{n-1}||...||m_1$ and get back $\sum(M_\alpha) = \sigma(m_n), \sigma(m_{n-1}), ..., \sigma(m_1)$.

Then, in order to forge M, A can just output the reverse of $\sum(M_\alpha)$, or $\sum(M) = \sigma(m_1), \sigma(m_2), ..., \sigma(m_n)$. M was never queried to the sign oracle, and so A has successfully forged M and so this scheme does not satisfy existential unforgeability.

3.2 Let A be our attacker that is given the public key. A can interact with the oracle $\text{Sign}_{sk}()$. By Kerckhoff's principle, A knows how the signing works. Then, in order to forge message $M = m_1||m_2||...||m_n$, A can interact with the sign oracle to sign messages $M_\alpha = m_1||m_2||...||m_{n-1}||0^k$, $M_\beta = 0^k||m_2||m_3||...||m_n$ and get back $\sum(M_\alpha) = \sigma(1||m_1), \sigma(2||m_2), ..., \sigma(n||0^k)$ and $\sum(M_\beta) = \sigma(1||0^k), \sigma(2||m_2), \sigma(3||m_3), ..., \sigma(n||m_n)$.

Then, in order to forge M, A can just replace the last σ of M_α with the last σ of M_β, namely replace $\sigma(n||0^k)$ with $\sigma(n||m_n)$. Then, we get $\sum(M) = \sigma(1||m_1), \sigma(2||m_2), ..., \sigma(n||m_n)$ which is a valid signature for M, but M was never queried to the sign oracle. Hence, A has successfully forged M and so this scheme does not satisfy existential unforgeability.

4a) Bob gets $m, \sigma = \text{Sign}_{sk}(m) = [f(m)^d \bmod N]$ where d is the private key. In order to verify that a message signature pair was indeed sent by Alice, Bob needs to compute σ^e and checks if $\sigma^e = [f(m) \bmod N]$ where e is Alice's public key. Note that Bob also needs access to this new encoding function f.

4b) Fix PPT attacker A, scheme π, message m_0

Define randomized experiment $\text{Forge}_{A,\pi,m_0}(n)$:

1. $pk, sk \leftarrow Gen(1^n)$
2. A given pk, interacts with oracle $\text{Sign}_{sk}(\cdot)$; let M be the set of messages sent to oracle, $m_0 \notin M$
3. A outputs (m_0, σ)
4. A succeeds and experiment evaluates to 1 if $\text{Vrfy}_{pk}(m_0, \sigma) = 1$, $m_0 \notin M$

π is target-message unforgeable if for all PPT attackers A, there is a negligible function ϵ such that:

$$\Pr[\text{Forge}_{A,\pi,m_0}(n) = 1] \leq \epsilon(n)$$

4c) The existential unforgeability definition is stronger than target-message since it states that a scheme is unforgeable for *any* message rather than a specific message. (existential unforgeability is target-message unforgeability for *all* messages)