Public Key Encryption

Q1: Explain semantic security vs. CPA-security for public key encryption schemes.

A1: In private key encryption schemes semantic security and CPA-security are different, but in public key setting they are the same, as the adversary knows about the public key it can make encryption queries itself.

Q2: Discuss the advantages and disadvantages of public-key encryption systems in comparison to symmetric-key encryption systems. Keep your comparison as detailed as possible, looking from multiple angles.

A2: Public key encryption schemes remove the need of holding the same key on both parties at the start. In other words, the efficiency of key distribution is better in public key cryptography. However, in terms of performance, symmetric key schemes are faster and more efficient. So usually, a key is distributed in public key setting at the start, and then use private key encryption afterwards.

Q3: Formally define the factoring assumption. Formally define the RSA assumption. Compare them, and explain the relationship between them.

A3: First, let us look at factoring assumption with a game as seen in algorithm 1. It is assumed that for algorithm 1 $\forall$ PPT $\mathcal{A}$, $\exists \texttt{negl}$

Algorithm 1 $\texttt{Fact}_{\mathcal{A},GenModulus}(n)$ game.

```
Chal computes (N, p, q) ← GenModulus(1^n).
Chal gives A only N.
A gives p' and q' to Chal.
if N = p'q' then
  A wins
end if
```

s.t. $\Pr[\mathcal{A} \text{ wins }] \leq \texttt{negl}(n)$.

For the RSA assumption, let us look at algorithm 2. It is assumed again that for algorithm 2 $\forall$ PPT $\mathcal{A}$, $\exists \texttt{negl}$ s.t. $\Pr[\mathcal{A} \text{ wins }] \leq \texttt{negl}(n)$.

They are closely related. In RSA, the attacked knows N, e but not d. It also has $c = x^e \bmod N$. Can it find x? Well, we may ask: how hard is it to compute e^{th} roots in modulo N? So far, the best known algorithm works in two steps:

Algorithm 2 $\texttt{Fact}_{\mathcal{A},GenModulus}(n)$ game.

$\texttt{Chal}$ computes $(N, e, d) \leftarrow GenRSA(1^n)$.
$\texttt{Chal}$ chooses $y \leftarrow \mathbb{Z}_N^*$.
$\texttt{Chal}$ gives N, e, y to $\mathcal{A}$.
$\mathcal{A}$ gives $x \in \mathbb{Z}_N^*$ to $\texttt{Chal}$.
if $x^e = y$ **then**
  $\mathcal{A}$ wins
end if

(1) Factor N. (*Hard*)
(2) Compute e^{th} roots in modulo p and q. (*Easy*)

As we can see, we actually are depending the hardness of factoring here!

Q4: Which standards should be employed when using RSA encryption scheme in practice? What are the proven security guarantees provided by each such standard?

A4: There are few RSA standards:

- **RSA PKCS v 1.5**: This one is not even CPA-secure! It is broken by the Bleichenbacher attack.
- **RSA-based KEM**: This one is CPA-secure, but not CCA-secure.
- **RSA-OAEP**: This one is actually CCA-secure, and it is the recommended one we should use.

Q5: What are the recommended minimum and maximum RSA key lengths these days? What is the effect of the key length on security? How should one determine the minimum and maximum key length to employ?

A5: Right now 2048 bits is the recommended key length, and in general between 1000-4000 should be used. Higher key lengths mean more security but less performance.

Q6: What is hybrid encryption? Why is it employed in practice? Why is just using public key encryption insufficient?

A6: Since private key encryption is faster, but public key encryption solves the key distribution problem, it is a good idea to use both together, which is what is done in hybrid encryption. Just using public

key encryption would be computational demanding, as it is less efficient.

Q7: What is the underlying security assumption for the ElGamal encryption system? Define it formally. Compare it with the underlying assumption of the Diffie-Hellman key exchange scheme. Which assumption is weaker (meaning that it is more likely to be true)?

A7: ElGamal is based on DDH (Decisional Diffie-Hellman Problem). The DDH can be described as: It is assumed again that for algorithm

Algorithm 3 DDH game.

`Chal` computes $(G, q, g) \leftarrow \mathcal{G}(1^n)$.
`Chal` chooses $x, y, z \leftarrow \mathbb{Z}_q$.
`Chal` picks a bit $b \leftarrow \{0, 1\}$.
`Chal` computes $h_1 = g^x$ and $h_2 = g^y$.
if $b = 1$ **then**
  `Chal` computes $h' = g^z$.
else
  `Chal` computes $h' = g^{xy}$.
end if
`Chal` gives h_1, h_2, h' to $\mathcal{A}$.
$\mathcal{A}$ returns a bit $b' \in \{0, 1\}$ to `Chal`.
if $b' = b$ **then**
  $\mathcal{A}$ wins.
end if

3 $\forall$ PPT $\mathcal{A}$, $\exists$`negl` s.t. Pr[$\mathcal{A}$ wins] $\leq$ `negl`(n).

The Diffie-Hellman key exchange was based on Discrete Logarithm problem, as described in algorithm 4. It is assumed again that for

Algorithm 4 D-Log game.

`Chal` computes $(G, q, g) \leftarrow \mathcal{G}(1^n)$.
`Chal` chooses $h \leftarrow G$.
`Chal` gives G, q, g, h to $\mathcal{A}$.
$\mathcal{A}$ returns $x \in \mathbb{Z}_q$ to `Chal`.
if $g^x = h$ **then**
  $\mathcal{A}$ wins.
end if

algorithm 4 $\forall$ PPT $\mathcal{A}$, $\exists$`negl` s.t. Pr[$\mathcal{A}$ wins] $\leq$ `negl`(n).

The relationship is that: if the discrete-logarithm problem is easy relative to some $\mathcal{G}$, then the CDH problem is too. Similarly, if CDH

is easy relative to $\mathcal{G}$, then so is the DDH problem. However, the converse is not true. There are groups where discrete-logarithm and CDH problems are believed to be hard even though DDH problem is easy.

If you have an algorithm that solves discrete-logarithm problem in some group, then you can easily construct one that can solve CDH and then DDH. For CDH, just compute y from h_2 and calculate $h' = h_1^{x_2}$. However, the other way around is not known. To solve DDH, just compute x from h_1 and y from h_2 and then check whether $h' = g^{xy}$.

This means, DDH is at least as easy as CDH, and CDH is at least as eays as DL. Or, the other way around, DL is at least as hard as CDH, and CDH is at least as hard as DDH.[1]. Thus, the hardness of DDH is a stronger assumption (is valid in fewer groups) than the hardness of CDH, which itself is still stronger than the hardness of DL.

For this question in particular, DDH is a stronger assumption than DL, thus may result in a weaker scheme as that assumption holds for less number of groups than DL.

Q8: Write down a full reduction proof for the textbook ElGamal encryption system.

A8: We would like to show that if DDH assumption holds, then ElGamal is CPA secure. We look at the contrapositive: if $\exists \mathcal{A}$ that breaks ElGamal, then we can construct B that breaks DDH assumption. For the sake of space, I am just writing the logic of the proof here. Let `Chal` be the challenger for B, and B is the challenger for $\mathcal{A}$. `Chal` gives B: $\langle G, q, g, h_1, h_2, h_3 \rangle$. Suppose $h_1 = g^x, h_2 = g^y$. B gives to A as public key $pk = \langle G, q, g, h_1 \rangle$. When $\mathcal{A}$ makes the challenge query m_0, m_1, B picks a bit $b \leftarrow \{0, 1\}$ and encrypts $c = \langle h_2, h_3 \cdot m_b \rangle$. For h_3, there are two cases:

- $h_3 = g^z$. In this case, the game is indistinguishable for $\mathcal{A}$, so it wins with probability $1/2$. $b' = 0$ with probability $1/2$.
- $h_3 = g^{xy}$. In this case, the ElGamal breaking $\mathcal{A}$ wins with probability $1/2 + \epsilon(n)$ where by the contrapositive statement $\epsilon(n)$ is non-negligible, so $b' = 1$ with probability $1/2 + \epsilon(n)$.

Suppose B outputs $b' = 1$ when $\mathcal{A}$ wins the game between them, $b' = 0$ otherwise. Also suppose that `Chal` chooses $b_c \in \{0, 1\}$, and when $b_c = 1$ it sets $h_3 = g^{xy}$, $h_3 = g^z$ otherwise.

[1] https://crypto.stackexchange.com/questions/1493/what-is-the-relation-between-discrete-log-computational-diffie-hellman-and-deci

Here, B wins with probability $1/2(1/2)+1/2(1/2+\epsilon) = 1/2+\epsilon(n)/2$. If $\epsilon(n)$ is non-negligible, this breaks DDH assumption, therefore $\epsilon(n)$ must be negligible, and thus ElGamal is CPA-secure.

Q9: Formally define TDP (trapdoor permutation) and TDF (trapdoor function). How does one use a TDP/TDF to obtain a public key encryption scheme?

A9: A trapdoor function family $\Pi = (Gen, f, Inv)$:

- Gen on input 1^n outputs $(I, \mathtt{td})$. Each value of I defines a set D_I that constitutes the domain and range of a permutation $f_I : D_I \to D_i$.
- Let $(I, \mathtt{td})$ be an output by $Gen(1^n)$. The deterministic inverting algorithm Inv, on input $\mathtt{td}$ and $y \in D_I$ outputs $x \in D_I$, denoted as $x = Inv_{\mathtt{td}}(y)$. It is required (except with negligible probability) that:

$$Inv_{\mathtt{td}}(f_I(x)) = x$$

Let $\Pi' = (Gen', f, Inv)$ be a TDP family with hard-core predicate $\mathtt{hc}$, then define a public key encryption scheme $\Pi = (Gen, Enc, Dec)$ as follows:

- Gen: On input 1^n, run $Gen'(1^n)$ to obtain $(I, \mathtt{td})$. Output the public key $pk = I$ and secret key $sk = \mathtt{td}$.
- Enc: On input a public key $pk = I$ and message (bit) $m \in \{0, 1\}$, choose a uniform $r \in D_I$ subject to the constraint $\mathtt{hc}_I(r) = m$. Output $c = f_I(r)$.
- Dec: On input a secret key $sk = \mathtt{td}$ and a ciphertext c, compute $r = Inv_I(c)$ and output $\mathtt{hc}_I(r)$.

Q10: How can one overcome the limitation that TDP/TDF is deterministic, while we know that deterministic encryption cannot be CPA- or CCA-secure?

A10: There is actually a bit of randomness, it is inside the encryption function when we choose a uniform $r \in D_I$ subject to the constraint $\mathtt{hc}_I(r) = m$. Thanks to this randomness, it is actually not an absolutely deterministic encryption.