Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Prof. Dennis Hofheinz
Julia Kastner

Solutions

Exercise 1. Prove that if $P = NP$ there does not exist a public key encryption (PKE) scheme which is IND-CPA secure.

Solution. Assume there exists a PKE scheme $E = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. We assume perfect correctness, i.e.

$$\forall m \in \mathcal{M}\colon \Pr\left[(\mathsf{pk}, \mathsf{sk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda); r \overset{\$}{\leftarrow} R\colon \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m; r)) = m\right] = 1$$

We define the following language of valid public-key-message-ciphertext tuples:

$$\mathcal{L} \coloneqq \{(\mathsf{pk}, m, c) \mid \exists r \in R\colon \mathsf{Enc}(\mathsf{pk}, m; r) = c\}$$

Clearly, $\mathcal{L} \in NP$ as there exists an efficiently checkable relation

$$R_{\mathcal{L}} = \{((\mathsf{pk}, m, c), r) \mid \mathsf{Enc}(\mathsf{pk}, m; r) = c\}$$

i.e. the encryption randomness r can be used as a witness.

As we also assumed that $P = NP$, there exists a polynomial-time Turing machine $\mathcal{T}$ that decides $\mathcal{L}$. An IND-CPA adversary $\mathcal{A}$ does the following: When it receives a public key pk it picks two random messages $m_0 \neq m_1 \in \mathcal{M}$ such that $|m_0| = |m_1|$. It submits the messages to its challenger and receives a ciphertext c^*. It then runs $\mathcal{T}$ on the tuple (pk, m_0, c^*) and outputs 0 if it accepts, 1 otherwise.

We note that as we assumed perfect correctness, it cannot happen that $(\mathsf{pk}, m_0, c^*) \in \mathcal{L}$ at the same time as $(\mathsf{pk}, m_1, c^*) \in \mathcal{L}$.

Alternative solution (that shows impossibility of even weaker security notions) Assume there exists a PKE scheme $E = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. We assume perfect correctness, i.e.

$$\forall m \in \mathcal{M}\colon \Pr\left[(\mathsf{pk}, \mathsf{sk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda); r \overset{\$}{\leftarrow} R\colon \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m; r)) = m\right] = 1$$

We note that each pk corresponds to a class of "equivalent" secret keys due to perfect correctness:

$$\forall m \in \mathcal{M}\colon \mathsf{Dec}(\mathsf{sk}_0, \mathsf{Enc}(\mathsf{pk}, m)) = m = \mathsf{Dec}(\mathsf{sk}_1, \mathsf{Enc}(\mathsf{pk}, m))$$

We can therefore define the following languages:

$$\mathcal{L}_{k,\mathsf{pk},\lambda} \coloneqq \left\{s \in \{0,1\}^k \middle| \exists (t, r) \in \{0,1\}^{B-k} \times R\colon (\mathsf{pk}, s \| t) = \mathsf{KeyGen}(1^\lambda; r)\right\}$$

where B is a bound on the size of sk which is polynomial in λas KeyGen has to be efficient. Informally speaking, this is the language of k-prefixes of sk that match pk. It is easy to see that (t, r) is a witness that a given s is in $\mathcal{L}_{k,\mathsf{pk},\lambda}$ and thus the language is in NP. As we further assumed that $P = NP$, there exist deterministic Turing machines $TM_{k,\mathsf{pk},\lambda}$ that decide $\mathcal{L}_{k,\mathsf{pk},\lambda}$ in polynomial time. We now do the following to find out the secret key corresponding to the public key pk: Start with $TM_{1,\mathsf{pk},\lambda}$ and run it on input 0. If it accepts, set $s_1 \coloneqq 0$, otherwise run $TM_{1,\mathsf{pk},\lambda}$ on input 1 and if it accepts set $s_1 = 1$. For each k (until we have computed a secret key sk of size up to B) we run $TM_{k,\mathsf{pk},\lambda}(s_{k-1}\|0)$ and if it accepts set $s_k := s_{k-1}\|0$, otherwise run $TM_{k,\mathsf{pk},\lambda}(s_{k-1}\|1)$ and if it accepts set $s_k := s_{k-1}\|1$. If neither TM accepts, output s_{k-1} as the secret key. When s_B is reached, output s_B as the secret key. It is easy to see that this algorithm outputs a valid secret key. This secret key can be used to break IND-CPA security of E by decrypting the challenge ciphertext.

Exercise 2. In the following, we want to consider how one can combine existing PKE schemes to build new ones.

(a) Consider two public-key encryption (PKE) schemes $E_i = (\mathsf{KeyGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i)$ for $i \in \{0,1\}$. Both E_0 and E_1 are correct and have the same message space $\mathcal{M}$. Assume only one of the schemes E_0 and E_1 is IND-CPA secure. Without knowing which scheme is secure, design a PKE scheme E_2 for $\mathcal{M}$ that uses E_0 and E_1 and is IND-CPA secure and provide a proof of IND-CPA security of your new scheme.

(b) Assume you have n PKE schemes $E_i = (\mathsf{KeyGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i)$ for $i \in \{1,\ldots,n\}$ with the same message space $\mathcal{M}$, all of which are correct, and at least one of which is IND-CPA secure (but you do not know which one). Use these schemes to construct a new scheme E_{n+1} that is IND-CPA secure. Provide a proof for the IND-CPA security of your new scheme.

Solution.

(a) We describe the scheme $E_2 = (\mathsf{KeyGen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$:

KeyGen_2**:** sample $(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KeyGen}_0(1^\lambda)$ and $(\mathsf{pk}_1, \mathsf{sk}_1) \xleftarrow{\$} \mathsf{KeyGen}_1(1^\lambda)$. Output $(\mathsf{pk}_2 = (\mathsf{pk}_0, \mathsf{pk}_1), \mathsf{sk}_2 = (\mathsf{sk}_0, \mathsf{sk}_1))$

Enc_2**:** For message m, sample $r \xleftarrow{\$} \{0,1\}^{|m|}$. Compute $c_0 \xleftarrow{\$} \mathsf{Enc}_0(\mathsf{pk}_0, r)$ and $c_1 \xleftarrow{\$} \mathsf{Enc}_1(\mathsf{pk}_1, r \oplus m)$. Output $c_2 := (c_0, c_1)$

Dec_2**:** compute $m_0 := \mathsf{Dec}_0(\mathsf{sk}_0, c_0)$ and $m_1 := \mathsf{Dec}_1(\mathsf{sk}_1, c_1)$. Output $m = m_0 \oplus m_1$.

Correctness follows from the correctness of E_0 and E_1 and from $m' = m_0 \oplus m_1 = r \oplus r \oplus m = m$. We briefly sketch the proof of IND-CPA security. First, assume E_0 is the IND-CPA secure scheme. A reduction from IND-CPA security of E_0 to IND-CPA security of E_2 receives a public key pk_0 of E_0 and samples a key pair $(\mathsf{sk}_1, \mathsf{pk}_1)$ using KeyGen_1. It ouputs $(\mathsf{pk}_0, \mathsf{pk}_1)$ to the adversary against IND-CPA security of E_2. When the adversary submits two challenge messages m_0, m_1, the reduction samples $r \xleftarrow{\$} \{0,1\}^{|m_0|}$ (note that a valid adversary submits messages of equal lengths). It submits its challenge messages $m'_0 = r$ adn $m'_1 = r \oplus m_0 \oplus m_1$. It further computes the ciphertext $c_1 \xleftarrow{\$} \mathsf{Enc}_1(\mathsf{pk}_1, m_0)$. When it receives a challenge ciphertext c' it sends (c', c_1) to the adversary. It forwards the output bit of the adversary to its own challenger. The reduction simulates the IND-CPA game perfectly to the adversary as in case $b = 0$, c', c_1 is a valid encryption of m_0, and in case $b = 1$ (c', c_1) is a valid encryption of m_1.

In the case that E_1 is the IND-CPA secure scheme we also provide a reduction. It receives the public key pk_1 from the challenger and samples its own key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$ using KeyGen_0. It outputs $(\mathsf{pk}_0, \mathsf{pk}_1)$ to the adversary. When the adversary submits challenge messages m_0, m_1, the reduction samples $r \xleftarrow{\$} \{0,1\}^{|m_0|}$ and submits the challenge messages $m'_0 = m_0 \oplus r$ and $m'_1 = m_1 \oplus r$ to the challenger to receive c'. It computes $c_0 \xleftarrow{\$} \mathsf{Enc}_0(\mathsf{pk}_0, r)$ and outputs (c_0, c') to the adversary. When the adversary outputs a bit b' it forwards b' to its own challenger.

(b) We extend our solution for subtask (a) to n values, i.e. we choose $n-1$ values $r_1, \ldots, r_{n-1}$ and compute $c_i \xleftarrow{\$} \mathsf{Enc}_i(\mathsf{pk}_i, r_i)$ for $i \in \{1, \ldots n-1\}$ and $c_n \xleftarrow{\$} \mathsf{Enc}_n(\mathsf{pk}_n, m \oplus r_1 \oplus \ldots \oplus r_{n-1})$. Decryption works in a straightforward manner and the proof of IND-CPA security is analogous to before.

Exercise 3. Consider a cyclic group $\mathbb{G}$ of known prime order $p > 2$ and let $\mathbf{g}$ be a generator of $\mathbb{G}$.

(a) You are given access to an oracle square which on input $\mathbf{g}^a$ outputs $\mathbf{g}^{(a^2)}$. Show that given access to square, there exists a polynomial-time algorithm that solves **DDH** in $\mathbb{G}$.

(b) You are given access to an oracle inv which on input $\mathbf{g}^a$ outputs $\mathbf{g}^{\frac{1}{a}}$. Show that given access to inv, there exists a polynomial-time algorithm that solves **DDH** in $\mathbb{G}$.

Solution.

(a) On input of $\mathbf{g}^x, \mathbf{g}^y, \mathbf{Z}$, we compute $\mathbf{g}^x \cdot \mathbf{g}^y = \mathbf{g}^{x+y}$ and submit it to the oracle square to obtain $\mathbf{g}^{x^2+2xy+y^2}$. We further submit $\mathbf{g}^x$ and $\mathbf{g}^y$ to obtain $\mathbf{g}^{x^2}$ and $\mathbf{g}^{y^2}$. We can compute $\mathbf{g}^{x^2+2xy+y^2}/\mathbf{g}^{x^2} = \mathbf{g}^{2xy+y^2}$ and $\mathbf{g}^{2xy+y^2}/\mathbf{g}^{y^2} = \mathbf{g}^{2xy}$. Compare $\mathbf{g}^{2xy} \stackrel{?}{=} \mathbf{Z} \cdot \mathbf{Z}$ to solve **DDH**.

(b) We use inv to implement square from subtask (a). Our implementation works as follows. On input $\mathbf{g}^a$ compute $\mathbf{g}^{a+1} = \mathbf{g}^a \cdot \mathbf{g}$. Use inv to compute $\mathbf{g}^{\frac{1}{a+1}}$. Furthermore compute $\mathbf{g}^{a-1} = \mathbf{g}^a/\mathbf{g}$ and use inv to obtain $\mathbf{g}^{\frac{1}{a-1}}$. We compute $\mathbf{g}^{\frac{2}{a^2-1}} = \mathbf{g}^{\frac{1}{a-1}} \cdot \mathbf{g}^{\frac{1}{a+1}}$. Use inv again to obtain $\mathbf{g}^{\frac{a^2-1}{2}}$. We can now compute $\mathbf{g}^{a^2-1} = \mathbf{g}^{\frac{a^2-1}{2}} \cdot \mathbf{g}^{\frac{a^2-1}{2}}$ and $\mathbf{g}^{a^2} = \mathbf{g}^{a^2-1} \cdot \mathbf{g}$. This yields the squaring oracle that can now be used as before.

Exercise 4. Consider two digital signature schemes $\Sigma_i = (\mathsf{KeyGen}_i, \mathsf{Sign}_i, \mathsf{Verify}_i)$ for $i \in \{0, 1\}$. You know that both of these signature schemes are correct, but only one of them is EUF-CMA secure. Using these two schemes, construct a new digital signature scheme Σ_2 that is EUF-CMA secure. Prove the security of your new scheme.

Solution. Our new scheme works as follows:

KeyGen_2 samples keys $(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KeyGen}_0(1^\lambda)$, $(\mathsf{pk}_1, \mathsf{sk}_1) \xleftarrow{\$} \mathsf{KeyGen}_1(1^\lambda)$, sets $\mathsf{pk}_2 = (\mathsf{pk}_0, \mathsf{pk}_1)$ and $\mathsf{sk}_2 = (\mathsf{sk}_0, \mathsf{sk}_1)$

$\mathsf{Sign}_2(\mathsf{sk}_2 = (\mathsf{sk}_0, \mathsf{sk}_1), m)$ Computes $\sigma_0 \xleftarrow{\$} \mathsf{Sign}_0(\mathsf{sk}_0, m)$ and $\sigma_1 \xleftarrow{\$} \mathsf{Sign}_1(\mathsf{sk}_1, m)$ and sets $\sigma_2 = (\sigma_0, \sigma_1)$

$\mathsf{Verify}_2(\mathsf{pk}_2 = (\mathsf{pk}_0, \mathsf{pk}_1), m, \sigma = (\sigma_0, \sigma_1))$ compute $b_0 = \mathsf{Verify}_0(\mathsf{sk}_0, m, \sigma_0)$ and $b_1 = \mathsf{Verify}_1(\mathsf{sk}_1, m, \sigma_1)$ and output $b = b_0 \wedge b_1$

We provide a reduction. Assume wlog scheme 0 is the EUF-CMA secure one. The reduction obtains pk_0 from the EUF-CMA challenger. It generates a key pair $(\mathsf{pk}_1, \mathsf{sk}_1) \xleftarrow{\$} \mathsf{KeyGen}_1(1^\lambda)$. Then it outputs $(\mathsf{pk}_0, \mathsf{pk}_1)$ to the adversary.

Whenever the adversary requests a signature on a message m, it asks the challenger for a signature on m under pk_0 and computes $\sigma_1 \xleftarrow{\$} \mathsf{Sign}_1(\mathsf{sk}_1, m)$. It then returns $\sigma = (\sigma_0, \sigma_1)$ to the adversary.

When the adversary outputs a message-signature pair $m^*, \sigma^* = (\sigma_0^*, \sigma_1^*)$ the reduction outputs m^*, σ_0^* to its challenger.

It is easy to see that the reduction perfectly simulates the EUF-CMA game to the adversary and that it wins the EUF-CMA game with the same probability as the adversary does.

References

[KL21] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. eng. Third edition. Chapman & Hall/CRC cryptography and network security. Boca Raton, Florida ; CRC Press, 2021. ISBN: 1-351-13303-9.