1 Ciphertext Collision in Semantically Secure Cryptosystems

We consider a public-key cryptosystem \((\text{Gen}, \mathcal{M}, \text{Enc}, \text{Dec})\). We assume perfect correctness, i.e. for all \(s\) and all \(x \in \mathcal{M}\), if \((K_p, K_s) \leftarrow \text{Gen}(1^s)\) then
\[
\Pr[\text{Dec}_{K_s}(\text{Enc}_{K_p}(x)) = x] = 1
\]

Given a probabilistic polynomial-time adversary \(A\), we consider the following game:

\begin{enumerate}
    \item \((K_p, K_s) \leftarrow \text{Gen}(1^s)\)
    \item \(X \leftarrow A(K_p)\)
    \item \(Y_0 \leftarrow \text{Enc}_{K_p}(X)\)
    \item \(Y_1 \leftarrow \text{Enc}_{K_p}(X)\)
    \item \(\text{return } 1\) if \(Y_0 = Y_1\)
\end{enumerate}

Q.1 Prove that if the cryptosystem is IND-CPA secure, then \(\Pr[\Gamma_A(s) \rightarrow 1]\) is negligible. Hint: construct an IND-CPA adversary with advantage related to \(\Pr[\Gamma_A(s) \rightarrow 1]\).

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{Algorithm } \(B(s)\): \\
1: receive \(K_p\) \\
2: run \(X_1 \leftarrow A(K_p)\) \\
3: pick \(X_0 \in \mathcal{M}\) such that \(X_0 \neq X_1\) \\
4: send \(X_0, X_1\), receive \(Y\) \\
5: \(Y' \leftarrow \text{Enc}_{K_p}(X_1)\) \\
6: \text{return } 1\) if \(Y = Y'\) \\
\hline
\end{tabular}
\end{center}

\textit{We define an IND-CPA adversary as follows:}

\textbf{Algorithm } \(B(s)\):
\begin{enumerate}
    \item receive \(K_p\)
    \item run \(X_1 \leftarrow A(K_p)\)
    \item pick \(X_0 \in \mathcal{M}\) such that \(X_0 \neq X_1\)
    \item send \(X_0, X_1\), receive \(Y\)
    \item \(Y' \leftarrow \text{Enc}_{K_p}(X_1)\)
    \item \text{return } 1\) if \(Y = Y'\)
\end{enumerate}

\textit{If } \(Y\) \textit{is the encryption of } \(X_1\), \textit{the IND-CPA game outputs } \(1\) \textit{with probability } \(\Pr[\Gamma_A(s) \rightarrow 1]\). \textit{If } \(Y\) \textit{is the encryption of } \(X_0\), \textit{we cannot have } \(Y = Y'\), \textit{so the game outputs } \(1\) \textit{with probability zero. Hence, the advantage of } \(B\) \textit{is exactly } \(\Pr[\Gamma_A(s) \rightarrow 1]\). \textit{Due to IND-CPA security, this is negligible.}
2 Non-Malleability in Adaptive Security

This exercise is inspired from Bellare-Desai-Pointcheval-Rogaway, Relations Among Notions of Security for Public-Key Encryption Schemes, CRYPTO 1998, LNCS vol. 1462, Springer.

We consider a public-key cryptosystem \((\text{Gen}, \mathcal{M}, \text{Enc}, \text{Dec})\). We assume perfect correctness, i.e. for all \(s\) and all \(x \in \mathcal{M}\), if \((K_p, K_s) \leftarrow \text{Gen}(1^s)\) then

\[
\Pr[\text{Dec}_{K_s}(\text{Enc}_{K_p}(x)) = x] = 1
\]

Given an adversary in two parts \(A = (A_1, A_2)\), a bit \(b \in \{0, 1\}\), and the security parameter \(s\), we define the IND-CCA game as follows:

**Game IND-CCA\(^b\)_A(s)**
1. \((K_p, K_s) \leftarrow \text{Gen}(1^s)\)
2. \((X_0, X_1, \sigma) \leftarrow A_1(\sigma)\) \(\triangleright \) \(\sigma\) is a “state” for \(A_1\) to transmit data to \(A_2\)
3. \(Y \leftarrow \text{Enc}_{K_p}(X_b)\)
4. \(b' \leftarrow A_2(\sigma, Y)\)
5. return \(b'\)

where the oracles \(\mathcal{O}_1\) and \(\mathcal{O}_2\) are defined as follows:

**Oracle \(\mathcal{O}_1(y)\):**
1. return \(\text{Dec}_{K_s}(y)\)
**Oracle \(\mathcal{O}_2(y)\):**
2. if \(y = Y\) then
3. abort the game
4. end if
5. return \(\text{Dec}_{K_s}(y)\)

We define the advantage

\[
\text{Adv}_{A}^{\text{IND-CCA}}(s) = \Pr[\text{IND-CCA}_A^1(s) \rightarrow 1] - \Pr[\text{IND-CCA}_A^0(s) \rightarrow 1]
\]

We say that the cryptosystem is IND-CCA secure if for all probabilistic polynomial time (PPT) adversary \(A\), \(\text{Adv}_{A}^{\text{IND-CCA}}(s)\) is negligible.

**Q.1** The definition of IND-CCA security which was given in the course (Def.5.5 on p.55–56 in the lecture notes, or slide p.404) was based on an interactive game between an adversary and a challenger. Prove that the two styles of definition for IND-CCA security are equivalent. (Carefully construct \((A_1, A_2)\) from an interactive adversary and an interactive adversary from \((A_1, A_2)\).)
In the interactive-style definition, the interactive adversary $A'$ receives a public key, makes decryption queries, submit two plaintexts, get a ciphertext, makes new decryption queries, and produces a bit. We define $A'$ from $(A_1, A_2)$ as follows:

**Algorithm $A'$**
1. wait until $K_p$ is received
2. simulate $A_1(K_p)$; any query to $O_1$ by this simulation is done by making decryption queries to the challenger
3. the simulation ends by producing $(X_0, X_1, \sigma)$
4. submit $(X_0, X_1)$ to the challenger and get $Y$ in return
5. simulate $A_2(\sigma, Y)$; any query to $O_2$ by this simulation is done by making decryption queries to the challenger
6. the simulation ends by producing $b'$
7. return $b'$

Clearly, the IND-CCA game with $(A_1, A_2)$ is perfectly simulated by the interactive game with $A'$. Hence, the advantages match.

Conversely, given an interactive adversary $A'$, we define $(A_1, A_2)$ as follows:

**Algorithm $A_1(K_p)$**
1. simulate $A'$ who starts by receiving $K_p$; any decryption query defines a query to $O_1$ and the simulated answer to query is made from the answer to the oracle
2. at some point, $A'$ issues $(X_0, X_1)$, we let $\sigma$ be the state of the simulation
3. return $(X_0, X_1, \sigma)$

**Algorithm $A_2(\sigma, Y)$**
4. resume the simulation of $A'$ from state $\sigma$, by starting from the reception of $Y$; any decryption query defines a query to $O_2$ and the simulated answer to query is made from the answer to the oracle
5. the simulation ends by releasing a bit $b'$
6. return $b'$

Again, the simulation is perfect. Hence, the advantages match.

Q.2 Let $A = (A_1, A_2)$ be an IND-CCA adversary. We define another IND-CCA adversary as follows:

**Algorithm $B_1^{O_1}(K_p)$**
1. simulate $A_1^{O_1}(K_p) \rightarrow (X_0, X_1, \sigma)$
2. if $X_0 = X_1$ then
   3. set $\sigma' \leftarrow (\sigma, 1)$
   4. pick an arbitrary $X$ such that $X \neq X_1$
   5. return $(X, X_1, \sigma')$
3. else
   6. set $\sigma' \leftarrow (\sigma, 0)$
   7. return $(X_0, X_1, \sigma')$
Algorithm $B_2^{O_2(\cdot)}(\sigma', Y)$
10: parse $\sigma' = (\sigma, c)$
11: if $c = 1$ then
12: return 0
13: else
14: simulate $A_2^{O_2(\cdot)}(\sigma, Y) \rightarrow b'$
15: return $b'$
16: end if

Prove that
$$\text{Adv}_{A}^{\text{IND-CCA}}(s) = \text{Adv}_{B}^{\text{IND-CCA}}(s)$$

Deduce that we can always assume $X_0 \neq X_1$ in an IND-CCA adversary.

Let $E$ be the event that $X_0 = X_1$ with adversary $A$. We have
$$\Pr[\text{IND-CCA}_A^1(s) \rightarrow 1|E] = \Pr[\text{IND-CCA}_A^0(s) \rightarrow 1|E]$$
thus
$$\text{Adv}_{A}^{\text{IND-CCA}}(s) = \Pr[\text{IND-CCA}_A^1(s) \rightarrow 1, -E] - \Pr[\text{IND-CCA}_A^0(s) \rightarrow 1, -E]$$

We have
$$\Pr[\text{IND-CCA}_B^b(s) \rightarrow 1] = \Pr[\text{IND-CCA}_A^b(s) \rightarrow 1, -E]$$

hence
$$\text{Adv}_{A}^{\text{IND-CCA}}(s) = \text{Adv}_{B}^{\text{IND-CCA}}(s)$$

We now define the NM-CCA game (for non-malleability) as follows:

**Game NM-CCA$_B^b(s)$**
1. $(K_p, K_s) \leftarrow \text{Gen}(1^s)$
2. $(M, \sigma) \leftarrow A_1^{O_1(\cdot)}(K_p)$ \quad $\sigma$ is a “state” which allows $A_1$ to transmit data to $A_2$
3. $X_0 \leftarrow M$ \quad $\triangleright M$ is a sampling algorithm defined by $A_1$
4. $X_1 \leftarrow M$ \quad $\triangleright$ we sample two independent plaintexts using $M$
5. $Y \leftarrow \text{Enc}_{K_p}(X_1)$
6. $(R, Y_1', \ldots, Y_n') \leftarrow A_2^{O_2(\cdot)}(\sigma, Y)$ \quad $\triangleright R$ is a poly. algo. returning a boolean
7. $X_i' \leftarrow \text{Dec}_{K_s}(Y_i')$, $i = 1, \ldots, n$
8. if $Y \not\in \{Y_1', \ldots, Y_n'\}$ and $\bot \not\in \{X_1', \ldots, X_n'\}$ and $R(X_b, X_1', \ldots, X_n')$ then
9:    return 1
10:    else
11:    return 0
12:   end if

We use the same oracles $\mathcal{O}_1$ and $\mathcal{O}_2$ as for IND-CCA. We define

$$\text{Adv}_{\mathcal{A}}^{\text{NM-CCA}}(s) = \Pr[\text{NM-CCA}^1_{\mathcal{A}}(s) \rightarrow 1] - \Pr[\text{NM-CCA}^0_{\mathcal{A}}(s) \rightarrow 1]$$

We say that the cryptosystem is NM-CCA secure if for all probabilistic polynomial time (PPT) adversary $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^{\text{NM-CCA}}(s)$ is negligible.

The goal of this exercise is to show the equivalence between NM-CCA security and IND-CCA security.

Q.3 We assume that $\mathcal{M}$ has a group structure (additively denoted), with at least two different elements 0 and 1, 0 being neutral. Assume that there is a polynomial algorithm $\text{Inc}$ such that for all $s$,

$$\Pr[\text{Dec}_{K_s}(\text{Inc}_{K_p}(\text{Enc}_{K_p}(X))) = X + 1] = 1$$

for $(K_p, K_s) \leftarrow \text{Gen}(1^s)$. By constructing an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, prove that the cryptosystem is not NM-CCA secure.

(The precision of the proof is important.)

HINT: use $M$ sampling in a set of two different plaintexts and $R$ defined by $R(X, X') = 1_{X' = X+1}$. 
Algorithm $A_1^{O_1}(K_p)$
1: pick $z, z' \in M$ such that $z \neq z'$
2: define $M$ sampling in $\{z, z'\}$ with uniform distribution
3: return $(M, K_p)$ ▷ we set $\sigma = K_p$

Algorithm $A_2^{O_2}(K_p, Y)$
4: $Y' \leftarrow \text{Inc}_{K_p}(Y)$
5: define $R$ by $R(X, X') = 1$ if $X' = X + 1$
6: return $(R, Y')$

Since $0 \neq 1$ in $M$, we have that $\text{Dec}_{K_p}(Y') = \text{Dec}_{K_p}(Y) + 1 \neq \text{Dec}_{K_p}(Y)$ so $Y' \neq Y$. We further have $\text{Dec}_{K_p}(Y') \neq \bot$. So, the outcome of the game is $\text{NM-CCA}^b_{A}(s) = R(X_b, \text{Dec}_{K_p}(Y')) = R(X_b, X_1 + 1) = 1_{X_1+1=X_b+1} = 1_{X_1=X_b}$ thanks to the group property.

In the NM-CCA game, if $M$ picks two identical plaintexts $X_0 = X_1$, then the outcome of the game is always 1 no matter what is $b$. If $X_0 \neq X_1$, the outcome of the game is $1_{b=1}$. Hence

$$\Pr[\text{NM-CCA}^1_{A}(s) \rightarrow 1] = 1$$

and

$$\Pr[\text{NM-CCA}^0_{A}(s) \rightarrow 1] = \frac{1}{2}$$

Therefore, we have

$$\text{Adv}_{A}^{\text{NM-CCA}}(s) = \frac{1}{2}$$

Q.4 Given an NM-CCA adversary $A = (A_1, A_2)$, we construct an IND-CCA adversary $B = (B_1, B_2)$ as follows:

Algorithm $B_1^{O_1}(K_p)$
1: simulate $A_1^{O_1}(K_p) \rightarrow (M, \sigma)$
2: sample $z_0 \leftarrow M$
3: sample $z_1 \leftarrow M$
4: set $\sigma' \leftarrow (z_0, z_1, \sigma)$
5: return $(z_0, z_1, \sigma')$

Algorithm $B_2^{O_2}(\sigma', Y)$
6: parse $\sigma' = (z_0, z_1, \sigma)$
7: simulate $A_2^{O_2}(\sigma, Y) \rightarrow (R, Y'_1, \ldots, Y'_n)$
8: for $i = 1, \ldots, n$ do
9: if $Y = Y'_i$ then return 0
10: $X'_i \leftarrow \mathcal{O}_2(Y'_i)$
11: if $X_i' = \bot$ then return $0$
12: end for
13: compute $b' \leftarrow R(z_1, X_1', \ldots, X_n')$
14: return $b'$

Prove that
\[
\text{Adv}_B^{\text{IND-CCA}}(s) = \text{Adv}_A^{\text{NM-CCA}}(s)
\]

Deduce that IND-CCA security implies NM-CCA security.

We first observe that since we check that $Y \neq Y_i'$, there is no problem to query $O_2(Y_i')$. By denoting $X_1 = z_b$ and $X_0 = z_{1-b}$, we can see that the IND-CCA$_B^b$ game is a perfect simulation of the NM-CCA$_A^b$ game (with some steps moved from the core game or adversary and decryption replaced by $O_2$). Hence

\[
\text{IND-CCA}_B^b = \text{NM-CCA}_A^b
\]

thus
\[
\text{Adv}_B^{\text{IND-CCA}}(s) = \text{Adv}_A^{\text{NM-CCA}}(s)
\]

Q.5 We assume that $\mathcal{M}$ has at least four elements.

Given an IND-CCA adversary $A = (A_1, A_2)$, we construct an NM-CCA adversary $B = (B_1, B_2)$ as follows:

**Algorithm $B_1^{O_1} (K_p)$**
1: simulate $A_1^{O_1} (K_p) \rightarrow (z_0, z_1, \sigma)$
2: define $M$ sampling in $\{z_0, z_1\}$ with uniform distribution
3: set $\sigma' \leftarrow (\sigma, K_p, z_0, z_1)$
4: return $(M, \sigma')$

**Algorithm $B_2^{O_2} (\sigma', Y)$**
5: parse $\sigma' = (\sigma, K_p, z_0, z_1)$
6: take an injective function $T$ on $\mathcal{M}$ such that $T(z_0) \notin \{z_0, z_1\}$ and $T(z_1) \notin \{z_0, z_1\}$
7: simulate $A_2^{O_2} (\sigma, Y) \rightarrow b'$
8: $Y' \leftarrow \text{Enc}_{K_p}(T(z_{b'}))$
9: define $R(X, X') = 1_{T(X) = X'}$
10: return $(R, Y')$

Prove that
\[
\text{Adv}_B^{\text{NM-CCA}}(s) = \frac{1}{2} \text{Adv}_A^{\text{IND-CCA}}(s)
\]

Deduce that NM-CCA security implies IND-CCA security.

**HINT**$_1$: assume without loss of generality that $z_0 \neq z_1$

**HINT**$_2$: compute $\Pr[X_0 = z_{b'}], \Pr[X_1 = z_{b'} | X_1 = z_1]$, and $\Pr[X_1 = z_{b'} | X_1 = z_0]$. 

7
Using Q.2, we can always transform the adversary to obtain \( z_0 \neq z_1 \). So, we assume \( z_0 \neq z_1 \) without loss of generality.

Due to correctness, we note that no decryption abort, so the outcome is

\[
\text{NM-CCA}_B^b(s) = 1_{R(X_b, \text{Dec}_K(Y))=1, Y \neq Y'} = 1_{R(X_b, T(z_{b'}))=1, Y \neq Y'} = 1_{T(X_b) = T(z_{b'}), Y \neq Y'}
\]

where \( Y \) is an encryption of \( X_1 \) and \( Y' \) is an encryption of \( T(z_{b'}) \). Given the assumptions on \( T \), we always have \( X_1 \neq T(z_{b'}) \). Due to the correctness of decryption, we deduce that we always have \( Y \neq Y' \). Due to injectivity, we deduce

\[
\Pr[\text{NM-CCA}_B^b(s) = 1] = \Pr[X_b = z_{b'}]
\]

Hence,

\[
\text{Adv}_{NM-CCA}^B(s) = \Pr[X_1 = z_{b'}] - \Pr[X_0 = z_{b'}]
\]

We have

\[
\text{Adv}_{NM-CCA}^B(s) = \Pr[X_1 = z_{b'}] - \Pr[X_0 = z_{b'}]
\]

Since \( b' \) only depends on \( X_1 \), \( X_0 \) is independent from \( z_{b'} \) so \( \Pr[X_0 = z_{b'}] = \frac{1}{2} \) (because \( z_0 \neq z_1 \)). Similarly, we have \( \Pr[X_1 = z_{b'}|X_1 = z_c] = \Pr[b' = c|X_1 = z_c] \) for \( c \in \{0, 1\} \). Thus, we have

\[
\Pr[X_1 = z_{b'}|X_1 = z_1] = \Pr[\text{IND-CCA}_A^1(s) = 1]
\]

\[
\Pr[X_1 = z_{b'}|X_1 = z_0] = 1 - \Pr[\text{IND-CCA}_A^0(s) = 1]
\]

Since \( \Pr[X_1 = z_0] = \Pr[X_1 = z_1] = \frac{1}{2} \),

\[
\Pr[X_1 = z_{b'}] = \frac{\Pr[\text{IND-CCA}_A^1(s) = 1] + 1 - \Pr[\text{IND-CCA}_A^0(s) = 1]}{2}
\]

Therefore

\[
\text{Adv}_{NM-CCA}^B(s) = \frac{1}{2} \left( \Pr[\text{IND-CCA}_A^1(s) = 1] - \Pr[\text{IND-CCA}_A^0(s) = 1] \right)
\]

Therefore

\[
\text{Adv}_{NM-CCA}^B(s) = \frac{1}{2} \text{Adv}_{IND-CCA}^A(s)
\]
3 Unruh Transform from $\Sigma$ to NIZK

This exercise is inspired from Unruh, Non-Interactive Zero-Knowledge Proofs in the Quantum Random Oracle Model, EUROCRYPT 2015, LNCS vol. 9057, Springer.

We consider a $\Sigma$ protocol $(P, V)$ for a relation $R$. We let $E$ be the set of challenges. Given some parameters $t$ and $m \geq 2$, we define the following non-interactive zero-knowledge proof (NIZK), with input $(x, w)$ such that $R(x, w)$ holds:

\begin{algorithm}
\textbf{Algorithm} Proof$(x, w)$:
1: \textbf{for} $i = 1$ to $t$ \textbf{do}
2: \hspace{1em} pick a sequence of fresh coins $\rho_i$
3: \hspace{1em} set $a_i \leftarrow P(x, w; \rho_i)$
4: \hspace{1em} \textbf{for} $j = 1$ to $m$ \textbf{do}
5: \hspace{2em} pick $e_{i,j} \in E - \{e_{i,1}, \ldots, e_{i,j-1}\}$ at random
6: \hspace{2em} set $z_{i,j} \leftarrow P(x, w, e_{i,j}; \rho_i)$
7: \hspace{2em} set $h_{i,j} \leftarrow G(z_{i,j})$
8: \hspace{1em} \textbf{end for}
9: \textbf{end for}
10: set $h \leftarrow H(x, (a_i, (e_{i,j}, h_{i,j})_{j=1,\ldots,m})_{i=1,\ldots,t})$
11: set $(J_1, \ldots, J_t) \leftarrow h$
12: set $z_i = z_{i, J_i}$ for $i = 1, \ldots, t$
13: set $\pi = (a_i, (e_{i,j}, h_{i,j})_{j=1,\ldots,m}, z_i)_{i=1,\ldots,t}$
14: \textbf{return} $\pi$
\end{algorithm}

This algorithm uses two random oracles $G$ and $H$. Oracle $H$ is assumed to return a $t$-tuple of integers between 1 and $m$. We use the following verification algorithm (with some missing step):

\begin{algorithm}
\textbf{Algorithm} Verify$(x, \pi)$:
1: parse $\pi = (a_i, (e_{i,j}, h_{i,j})_{j=1,\ldots,m}, z_i)_{i=1,\ldots,t}$
2: set $h \leftarrow H(x, (a_i, (e_{i,j}, h_{i,j})_{j=1,\ldots,m})_{i=1,\ldots,t})$
3: set $(J_1, \ldots, J_t) \leftarrow h$
4: verify \ldots
5: verify $V(x, a_i, e_{i,j}, z_i)$ for $i = 1, \ldots, t$
6: verify $h_{i,J_i} = G(z_i)$ for $i = 1, \ldots, t$
7: \textbf{return} 1 if all verifications passed
\end{algorithm}

Q.1 By taking the verification with the missing step, give an algorithm to forge a proof given $x$ but without the knowledge of $w$.

Which step should be added to have a sound proof?
We use the simulator of the $\Sigma$ protocol and all $e_{i,j}$ equal:

**Algorithm** Forge\((x)\):
1: pick $e \in E$ at random
2: \((a, e, z) \leftarrow S(x, e)\)
3: set $a_i = a$ for $i = 1, \ldots, t$
4: set $e_{i,j} = e$ for $i = 1, \ldots, t$, $j = 1, \ldots, m$
5: set $z_{i,j} = z$ for $i = 1, \ldots, t$, $j = 1, \ldots, m$
6: set $h_{i,j} = G(z)$ for $i = 1, \ldots, t$, $j = 1, \ldots, m$
7: set $h \leftarrow H(x, (a_i, (e_{i,j}, h_{i,j}))_{j=1,\ldots,m})_{i=1,\ldots,t}$
8: set $(J_1, \ldots, J_t) \leftarrow h$
9: set $z_i = z_{i,J_i}$ for $i = 1, \ldots, t$
10: set $\pi = (a_i, (e_{i,j}, h_{i,j}))_{j=1,\ldots,m}, z_i)_{i=1,\ldots,t}$
11: return $\pi$

It is clear that the output $\pi$ passes the verification with the missing step.
The missing step is
1: for $i = 1$ to $t$ do
2: verify that $e_{i,1}, \ldots, e_{i,m}$ are pairwise different
3: end for

**Q.2** With the new verification step from the last question, give an algorithm with complexity $O(m^t)$ to forge a valid $\pi$ from $x$ but without $w$. 

We try to predict the index of the challenges which will be verified and use the simulator of the $\Sigma$ protocol. We proceed as follows:

**Algorithm** $\text{Forge}(x)$:

1. repeat
2. for $i = 1$ to $t$ do
3. pick $J_i \in \{1, \ldots, m\}$
4. pick $e_{i,J_i} \in E$ at random
5. $(a_i, e_{i,J_i}, z_i) \leftarrow S(x, e_{i,J_i})$
6. for $j = 1$ to $m$ do
7. if $j \neq J_i$ then
8. pick $e_{i,j} \in E - \{e_{i,1}, \ldots, e_{i,j-1}, e_{i,J_i}\}$ at random
9. set $z_{i,j}$ at random
10. set $h_{i,j} \leftarrow G(z_{i,j})$
11. end if
12. end for
13. end for
14. set $h \leftarrow H(x, (a_i, (e_{i,j}, h_{i,j})_{j=1,\ldots,m})_{i=1,\ldots,t})$
15. until $(J_1, \ldots, J_t) = h$
16. set $z_i = z_{i,J_i}$ for $i = 1, \ldots, t$
17. set $\pi = (a_i, (e_{i,j}, h_{i,j})_{j=1,\ldots,m}, z_i)_{i=1,\ldots,t}$
18. return $\pi$

Since $h$ randomly picks $J_1, \ldots, J_t$, each iteration succeeds with probability $m^{-t}$. Hence, we need $\mathcal{O}(m^t)$ iterations until we succeed.

Q.3 Construct a simulator in the random oracle model to show that the protocol is non-interactive zero-knowledge.
Algorithm \textit{Simulate}(x):
\begin{algorithmic}[1]
\FOR{$i = 1$ \textbf{to} $t$}
\STATE pick $J_i \in \{1, \ldots, m\}$
\STATE pick $e_{i,J_i} \in E$ at random
\STATE $(a_i, e_{i,J_i}, z_i) \leftarrow S(x, e_{i,J_i})$
\STATE set $h_{i,J_i} \leftarrow G(z_i, J_i)$ \Comment{simulate $G$}
\FOR{$j = 1$ \textbf{to} $m$}
\IF{$j \neq J_i$}
\STATE pick $e_{i,j} \in E \setminus \{e_{i,1}, \ldots, e_{i,j-1}, e_{i,J_i}\}$ at random
\STATE set $h_{i,j}$ at random
\ENDIF
\ENDFOR
\STATE set $h \leftarrow (J_1, \ldots, J_m)$
\STATE set $h = H(x, (a_i, (e_{i,j}, h_{i,j})_{j=1,\ldots,m})_{i=1,\ldots,t})$ \Comment{simulate $H$}
\STATE set $z_i = z_{i,J_i}$ for $i = 1, \ldots, t$
\STATE set $\pi = (a_i, (e_{i,j}, h_{i,j})_{j=1,\ldots,m}, z_i)_{i=1,\ldots,t}$
\STATE \textbf{return} $\pi$
\ENDFOR
\end{algorithmic}

We should show that a limited distinguisher receiving $\pi$ and playing with the simulator of $G$ and $H$ cannot distinguish this $\pi$ from a genuine one. For this, we should argue that it cannot find the correct $z_{i,j}$ for $j \neq J_i$, except with negligible probability (because he would, together with $a_i$, $e_{i,J_i}$, and $z_{i,J_i}$, be able to extract a witness with the $\Sigma$ extractor, which is assumed to be hard). Without being able to query $G$ with the right $z_{i,j}$, the value $h_{i,j}$ is free. Thus, the distinguisher cannot see if $h_{i,j}$ was randomly selected by the simulator without knowing $z_{i,j}$ or randomly selected by $G$.

Q.4 Let $P^*(x)$ be an algorithm taking $x$ as input, interacting with $G$ and $H$, and forging a valid $\pi$ with probability $p$. Use the next questions to prove that there is an extractor who can run $P^*$ once to extract a witness $w$ for $x$ with probability at least $p - \text{negl}$.

Q.4a Transform $P^*$ into an algorithm $P'$ who either aborts or makes a valid $\pi$. It returns $\pi$ with probability $p$, and a complexity similar to $P^*$.

The algorithm $P'(x)$ first runs $P^*(x)$ and obtain $\pi$. Then, it parses $\pi = (a_i, (e_{i,j}, h_{i,j})_{j=1,\ldots,m}, z_i)_{i=1,\ldots,t}$ and runs $\text{Verify}(x, \pi)$. If verification fails, $P'$ aborts. Otherwise, it returns $\pi$. Clearly, the probability of success is the same and the complexity is similar.

Note that $P'$ always queries $H(x, (a_i, (e_{i,j}, h_{i,j})_{j=1,\ldots,m})_{i=1,\ldots,t}) = (J_1, \ldots, J_t)$. If also queries $G(z_i) = h_{i,J_i}$ for every $i$. 

12
Q.4b Construct an extractor $E$ on the previous $P'$ such that by observing only one execution of $P'$ with all queries to $G$ and $H$, either $P'$ aborts, or $E$ finds a witness for $x$, or $E$ aborts. But the probability that $E$ aborts is bounded by $n_Gn_HmtN^{-1} + n_Hm^{-t}$, where $n_G$ is the number of queries to $G$, $n_H$ is the number of queries to $H$, and $N$ is the size of the range of $G$.

Hint: say that a query $q$ to $H$ is good if it can be parsed in the form

$$q = x, (a_i, (e_{i,j}, h_{i,j})_{j=1,...,m})_{i=1,...,t}$$

Consider an extractor which aborts if any fresh query to $G$ returns a value $h_{i,j}$ which is included in a previous good query $q$ to $H$. Define another abort condition and extract a witness in remaining cases.

---

We consider an execution of $P'$ with all its queries to $G$ and $H$. Let $q$ be a fresh query to $H$ by $P'$. We say that $q$ is a good fresh query if $q$ parses into some $q = x, (a_i, (e_{i,j}, h_{i,j})_{j=1,...,m})_{i=1,...,t}$ such that for every $i$, all $e_{i,j}$ are pairwise different. So, $q$ defines a sequence of $a_i$, arrays of $e_{i,j}$ and $h_{i,j}$.

Note that if $P'$ succeeds to forge a valid $\pi$, there must be a good fresh query $q$ which matches the content of $\pi$.

If any fresh query to $G$ after the query $q$ to $H$ returns one of the values $h_{i,j}$ (there are $mt$ of them), the extractor aborts. So, the probability to abort for this case is bounded by $n_Gn_HmtN^{-1}$.

For each good fresh $q$, we define $J_q(i)$ as the set of $j$ such that $h_{i,j}$ was returned by $G$ at some point in the past. (Note that unless the extractor aborts, there won't be any future query to $G$ returning $h_{i,j}$.) We let $J'_q(i)$ be the subset of $J_q(i)$ such that there exists one query $q'$ to $G$ which returned $h_{i,j}$ and satisfying the condition $V(x, a_i, e_{i,j}, z_{i,j})$. If there is any $i$ such that $J'_q(i)$ has at least two elements, we can use the extractor to get a witness for $x$.

Now, we consider the case where for all $i$, $J'_q(i)$ has at most one element. When $H$ returns $(J_1, \ldots, J_t)$ to the fresh query $q$, if we have that $J_i \notin J'_q(i)$ for all $i$, then we make the extractor abort. Clearly, the probability this happens is bounded by $m^{-t}$. Applying this to all queries to $H$, the probability to abort is bounded by $nm^{-t}$.

If the extractor does not abort and $P'$ succeeds to make a valid $\pi$, we note that there is a good query $q$ to $H$ made by the verification. We take the fresher query equal to $q$. We also note that for all $i$, the verification in $P'$ makes a query $G(z_i) = h_{i,J_i}$ for each $i$. So, $z_i$ cannot be a fresh query and we must have $J_i \in J'_q(i)$ for all $i$. Hence, either $E$ succeeded to extract a witness or it aborted on that fresh good query.