CS 276 – Cryptography Oct 8, 2014

Lecture 11: Non-Interactive Zero-Knowledge II

Instructor: Sanjam Garg *Scribe: Rafael Dutra*

1 Non-Interactive Zero-Knowledge in the Hidden-Bits Model for the Graph Hamiltonian problem

Definition 1 *A Hamiltonian cycle in a graph is a cycle that visits each vertex exactly once. A Hamiltonian graph is a graph that contains a Hamiltonian cycle. More precisely, given a graph $G = (V, E)$ with $|V| = n$, we say that G is a Hamiltonian graph if there are $x_1, \ldots, x_n \in V$ such that $\forall i \in \{1, \ldots, n-1\} : (x_i, x_{i+1}) \in E$ and $(x_n, x_1) \in E$.*

It is well known that the problem of determining if a graph is Hamiltonian is NP-complete. Here we will construct a NIZK proof in the hidden-bits model (HBM) that is able to prove that a graph is Hamiltonian.
First we define how graphs are represented as matrices.

Definition 2 *A graph $G = (V, E)$ with $|V| = n$, can be represented as a $n \times n$ adjacency matrix M_G of boolean values such that* $M_G[i,j] = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases}$
A cycle matrix is a matrix which corresponds to a graph that contains a Hamiltonian cycle and contains no edges outside this cycle.
A permutation matrix is a boolean matrix such that each row and each column has exactly one entry equal to 1.

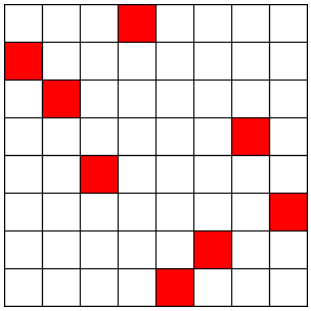

Figure 1: Cycle matrix.

Every cycle matrix is a permutation matrix, but the converse is not true. For each size n, there are $n!$ different permutation matrices but only $(n-1)!$ cycle matrices.
In Figure 1, one can see the cycle matrix as a cycle $(1, 4, 7, 6, 8, 5, 3, 2)$ on the set $\{1, 2, 3, 4, 5, 6, 7, 8\}$. In Figure 2, it is possible to interpret the matrix as a permutation $(1)(2, 8, 6, 5)(3, 7, 4)$ on the same set.

Theorem 1 *There is a non-interactive zero-knowledge (NIZK) proof in the hidden-bits model (HBM) for the problem of proving that a graph is Hamiltonian.*

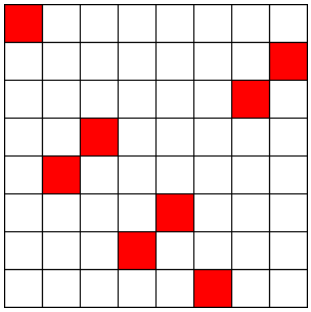

Figure 2: Permutation matrix.

Proof. In the hidden-bits model (HBM), there is a random string r with l bits that the prover can read. The prover should be able to produce a proof ϕ and choose a set $I \subseteq \{1, 2, \ldots, l\}$ such that the proof and the bits of the string corresponding to the set I will be revealed to the verifier.

$$P \xrightarrow{\phi, I, \{r_i \mid i \in I\}} V$$

The objective is to convince the verifier that the assertion is correct (the graph G is Hamiltonian). Let the graph be $G = (V, E)$ with $|V| = n$.
Suppose we know for some reason that the random string r comes from a distribution such that this string represents the entries from a $n \times n$ cycle matrix M_c. Then a proof can be produced as follows.
Since the prover P knows the Hamiltonian cycle $x_1, \ldots, x_n$ in G, he can find a function $\phi : V \to \{1, 2, \ldots, n\}$ that puts the Hamiltonian cycle exactly over the cycle of M_c. More precisely, for this function we have $M_c[\phi(x_i), \phi(x_{i+1})] = 1$ for each edge (x_i, x_{i+1}) in the Hamiltonian cycle of G (we view indices modulo n).
This means that all the edges of M_c will be covered by edges of G. Conversely, all the non-edges of G must be taken to non-edges of M_c.
So the strategy for the prover is to reveal the mapping ϕ and also reveal that $\phi(e)$ is a non-edge for all the non-edges e of G. More precisely, for the set $I = \{(\phi(u), \phi(v)) \mid (u, v) \in V \times V \setminus E\}$, P reveals $M_c[\phi(u), \phi(v)] = 0$, which proves that $(\phi(u), \phi(v))$ is a non-edge of M_c.

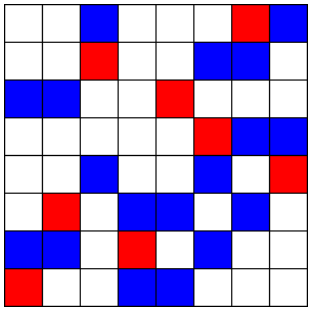

Figure 3: Graph matrix that includes a Hamiltonian cycle. Edges are blue/red and the cycle is red. White cells are non-edges.

A visual example is shown in Figure 3. The cycle graph M_c given by the random string corresponds

to the red cells. These cells have value 1 in the matrix M_c and all other cells have value 0. The prover P provides a bijection ϕ that sends the edges of G to this matrix in such a way that all red cells are covered and some others may also be covered (blue cells). The important property guaranteed is that all the non-edges of G are sent to cells that have a value 0 in the matrix (white cells).
This proof satisfies the three properties required by a zero knowledge proof.
Completeness: if P and V are both honest, then P will be able to convince V that the statement is true. That's because P knows the Hamiltonian cycle of G, hence he is always able to produce the mapping ϕ.
Soundness: if P is lying and trying to prove a false statement, then he will get caught with probability 1. If P does not know any Hamiltonian cycle in G, then any function ϕ he chooses will not cover all the cycle in M_c. Hence there will be an entry in the matrix M_c which is one and will be revealed as a non-edge of G.
Zero Knowledge: V cannot get any information besides the fact that P knows a Hamiltonian cycle in G. A simulator S for this proof can be simply a machine that generates a random permutation $\phi : V \to \{1, 2, \ldots, n\}$ and reveals zeros for all the non-edges of $\phi(G)$.

In this proof we assumed the random string r comes from a very specific distribution that corresponds to cycle matrices. Now we need to show that the general problem (where r comes from a random uniform distribution of l bits) can be reduced into this previous scenario.
We proceed as follows. Let the length of the random string be $l = \lceil 3 \cdot \log_2 n \rceil \cdot n^4$. We see the random string r as n^4 blocks of $\lceil 3 \cdot \log_2 n \rceil$ bits and we generate a random string r' of length n^4 such that each bit in r' is 1 if and only if all the bits is the corresponding block of r are equal to 1. This way, the probability that each bit in r' is 1 is $\Pr[r'_i = 1] \approx \frac{1}{n^3}$.
Then we create a $n^2 \times n^2$ matrix M whose entries are given by the bits of r'. Let x be the number of entries 1 in the matrix M. The expected value for x is $\frac{n^4}{n^3} = n$. And the probability that x is exactly n is noticeable. To prove that, we can use Chebyshev's inequality.
We have

$$\Pr[|x - n| \geq n] \leq \frac{\sigma^2}{n^2} = \frac{n^4 \cdot \frac{1}{n^3} \cdot \left(1 - \frac{1}{n^3}\right)}{n^2} < \frac{1}{n}$$

So we have $\Pr[1 \leq x \leq 2n - 1] > \frac{n-1}{n}$. But the probability $\Pr[x = k]$ is maximal for $k = n$, so we conclude that $\Pr[x = n] > \frac{n-1}{n(2n-1)} > \frac{1}{3n}$.
Now suppose that this event $(x = n)$ occurred and we have exactly n entries equal to 1 in matrix M. What is the probability that those n entries are all in different rows and are all in different columns?
We can think about the problem this way: after k entries 1 have been added to the matrix, the probability that a new entry will be in a different row and different column is given by $\left(1 - \frac{k}{n^2}\right)^2$. Multiplying all these values we get

$$\begin{aligned}\Pr[\text{no collision}] &\geq \left(1 - \frac{1}{n^2}\right)^2 \cdot \left(1 - \frac{2}{n^2}\right)^2 \cdots \left(1 - \frac{n-1}{n^2}\right)^2 \\ &> 1 - 2\left(\frac{1}{n^2} + \frac{2}{n^2} + \cdots + \frac{n-1}{n^2}\right) = 1 - \frac{n-1}{n} > \frac{1}{n}.\end{aligned}$$

Now assume that this event happened: the matrix M has exactly n entries equal to 1 and they are all in different rows and different columns. Then we can define a new $n \times n$ matrix M_c by selecting only those n rows and n columns of M. By construction, M_c is a permutation matrix.
The probability that a permutation matrix is a cycle matrix is $\frac{(n-1)!}{n!} = \frac{1}{n}$.
Now let's join all those probabilities. The probability that this construction works fine is at least

$$\frac{1}{3n} \cdot \frac{1}{n} \cdot \frac{1}{n} > \frac{1}{3n^3},$$

because this is the probability that M has exactly n ones, there are no collisions of those ones in any row or column and the $n \times n$ matrix constructed with them is a cycle matrix.
So we conclude that there is a noticeable probability that this construction will generate a cycle matrix. If we repeat this process n^4 times, the probability that this construction will work at least once is at least

$$1 - \left(1 - \frac{1}{3n^3}\right)^{n^4} \approx 1 - e^{-\frac{n}{3}} = 1 - negl(n).$$

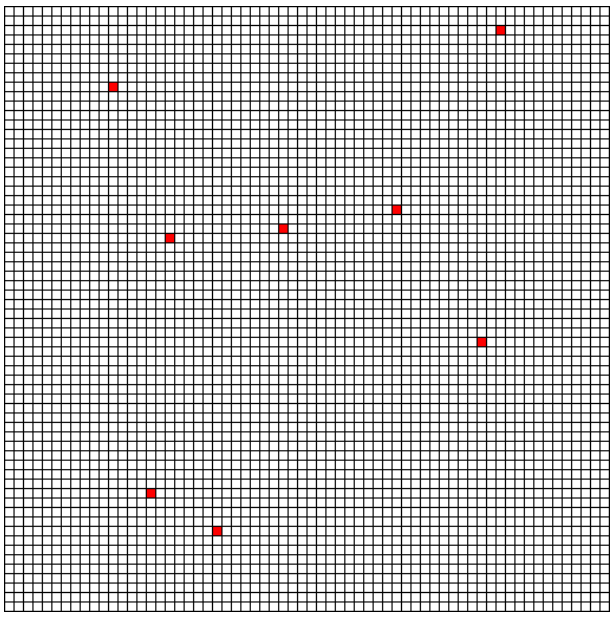

Figure 4: Matrix M which is $n^2 \times n^2$ for $n = 8$.

An example is shown in Figures 4 and 5.
So the proof systems works as follows. Given a random string r, the prover P tries to execute the construction above to obtain a cycle matrix. If the construction fails, the prover simply reveals all the bits in the string r to the verifier, who checks that the constructions indeed fails. If the construction succeeds, the prover reveals all the entries in the random string r that correspond to values in the matrix M which are not used in matrix M_c. The verifier will check that all these values for matrix M are indeed 0.
Then the prover proceeds as in the previous scenario using matrix M_c: he reveals the transformation ϕ and opens all the non-edges.

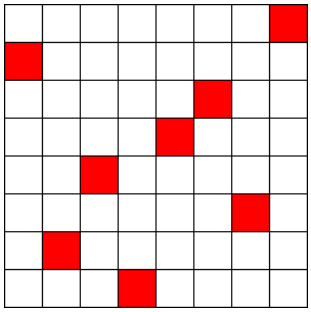

Figure 5: Matrix M_c which is $n \times n$ for $n = 8$. The construction worked, because M_c is a cycle matrix.

This process is repeated n^4 times. Or, equivalently, a big string of length $\lceil 3 \cdot \log_2 n \rceil \cdot n^4 \cdot n^4$ is used and they are all executed together. This produces a zero knowledge proof.
Completeness: if P knows the Hamiltonian cycle of G, then he will be able to find a suitable transformation ϕ whenever a cycle graph is generated by the construction.
Soundness: if P is lying and trying to prove a false statement, then he will get caught with very high probability. If any of the n^4 iterations produces a cycle graph, then P will be caught. So the probability that he will be caught is $1 - e^{-\frac{n}{3}} = 1 - negl(n)$.
Zero Knowledge: again V cannot get any information if the construction works. And if the construction doesn't work, all V gets is the random string r, which also doesn't give any information. ■

Theorem 2 *For any language L in NP, there is a non-interactive zero-knowledge (NIZK) proof in the hidden-bits model (HBM) for the language L.*

Proof. The language L^* of Hamiltonian graphs is NP-complete. So any problem in L can be reduced to a problem in L^*. More precisely, there is a polynomial-time function f such that

$$x \in L \iff f(x) \in L^*.$$

So given an input x, the prover can simply calculate $f(x)$ and produce a NIZK proof in the hidden-bits model for the fact that $f(x) \in L^*$. Then the verifier just needs to calculate $f(x)$ and check if the proof for the fact $f(x) \in L^*$ is correct. ■

Theorem 3 *For any language L in NP, there is a non-interactive zero-knowledge (NIZK) proof in the common reference string (CRS) model for the language L.*

Proof. On the previous lecture, it was shown that any NIZK proof in the hidden-bits model can be converted into a NIZK proof in the standard (common reference string) model by using a trapdoor permutation. ■

2 Chosen Ciphertext Attack for Public Key Encryption

Definition 3 *A public key encryption scheme (PKE) given by the three efficient procedures (G, E, D) is IND-CPA-secure if no adversary A has a significant advantage in the game represented in Table 1.*

Challenger		Adversary
$(P_k, S_k) = G(1^k)$	$\xrightarrow{P_k}$	
	$\xleftarrow{m'_0, m'_1}$	
$b \xleftarrow{\$} \{0,1\}$		
$c^* = E(P_k, m'_b)$	$\xrightarrow{c^*}$	
	$\xleftarrow{b'}$	
Output 1 if $(b' = b)$		

Table 1: CPA security.

This means that every probabilistic polynomial-time (PPT) adversary A has only a negligible advantage (over a random guess) when guessing which message (m'_0 or m'_1) the challenger used to obtain c^*.
Note that, since the adversary has the public key P_k, he is able to encrypt any polynomial number of plaintexts during the game.

Definition 4 *A public key encryption scheme (PKE) given by the three efficient procedures (G, E, D) is IND-CCA1-secure if no adversary A has a significant advantage in the game represented in Table 2.*

Challenger		Adversary
$(P_k, S_k) = G(1^k)$	$\xrightarrow{P_k}$	
	$\xleftarrow{c_1}$	
$m_1 = D(S_k, c_1)$	$\xrightarrow{m_1}$	
	$\vdots$	
	$\xleftarrow{c_q}$	
$m_q = D(S_k, c_q)$	$\xrightarrow{m_q}$	
	$\xleftarrow{m'_0, m'_1}$	
$b \xleftarrow{\$} \{0,1\}$		
$c^* = E(P_k, m'_b)$	$\xrightarrow{c^*}$	
	$\xleftarrow{b'}$	
Output 1 if $(b' = b)$		

Table 2: Non-adaptive CCA security.

In this definition the adversary may send some polynomial number of queries to be decrypted before he receives the challenge ciphertext c^*.

Definition 5 *A public key encryption scheme (PKE) given by the three efficient procedures (G, E, D) is IND-CCA2-secure if no adversary A has a significant advantage in the game represented in Table 3.*

Challenger		Adversary
$(P_k, S_k) = G(1^k)$	$\xrightarrow{P_k}$	
	$\xleftarrow{c_1}$	
$m_1 = D(S_k, c_1)$	$\xrightarrow{m_1}$	
	$\vdots$	
	$\xleftarrow{c_q}$	
$m_q = D(S_k, c_q)$	$\xrightarrow{m_q}$	
	$\xleftarrow{m'_0, m'_1}$	
$b \xleftarrow{\$} \{0,1\}$		
$c^* = E(P_k, m'_b)$	$\xrightarrow{c^*}$	
	$\xleftarrow{c_{q+1}}$	$c_{q+1} \neq c^*$
$m_{q+1} = D(S_k, c_{q+1})$	$\xrightarrow{m_{q+1}}$	
	$\vdots$	
	$\xleftarrow{c_{q+l}}$	$c_{q+l} \neq c^*$
$m_{q+l} = D(S_k, c_{q+l})$	$\xrightarrow{m_{q+l}}$	
	$\xleftarrow{b'}$	
Output 1 if $(b' = b)$		

Table 3: Adaptive CCA security.

Note that, in this adaptive version, the adversary is able to send more queries to the challenger even after having seen the challenge ciphertext c^*. The only thing we require is that he does not pass the challenge ciphertext c^* itself for those queries.

Theorem 4 *Given an IND-CPA-secure public key encryption scheme (G, E, D), it is possible to construct an IND-CCA1-secure public key encryption scheme (G', E', D').*

Proof. Let $(P_{k1}, S_{k1}) \leftarrow G(1^k)$, $(P_{k2}, S_{k2}) \leftarrow G(1^k)$ be two pairs of keys generated by the IND-CPA scheme (G, E, D). We claim that there is a NIZK proof system that is able to prove that c_1 and c_2 are ciphertexts obtained by the encryption of the same message m under the keys P_{k1} and P_{k2}, respectively.
More precisely, we claim that there is a NIZK proof system for the language

$$L = \{(c_1, c_2) \mid \exists r_1, r_2, m \text{ such that } c_1 = E(P_{k1}, m; r_1) \text{ and } c_2 = E(P_{k2}, m; r_2)\},$$

where $E(P_{ki}, m; r_i)$ represents the output of $E(P_{ki}, m)$ when the random coin flips of the procedure E are given by r_i.
The language L is clearly in NP, since for every $x = (c_1, c_2) \in L$ there is a witness $w = (r_1, r_2, m)$ that proves that $x \in L$. Given x and w, there is an efficient procedure to verify if w is a witness to the fact that $x \in L$.
By Theorem 3, there is a NIZK proof system for any language in NP, so our claim holds. Let this NIZK proof system be given by the procedures (K, P, V). We can assume that this is an adaptive

NIZK, because it is always possible to construct an adaptive NIZK from a non-adaptive NIZK proof system. Then we define our public key encryption scheme (G', E', D') as follows:

$$
\begin{aligned}
G'(1^k): \quad & (P_{k1}, S_{k1}) \leftarrow G(1^k) \\
& (P_{k2}, S_{k2}) \leftarrow G(1^k) \\
& \sigma \leftarrow K(1^k) \\
& \text{let } P'_k = (P_{k1}, P_{k2}, \sigma) \text{ and } S'_k = (S_{k1}) \text{ in} \\
& \text{return } (P'_k, S'_k)
\end{aligned}
$$

$$
\begin{aligned}
E'(P'_k, m): \quad & \text{let } (P_{k1}, P_{k2}, \sigma) = P'_k \text{ in} \\
& c_1 \leftarrow E(P_{k1}, m; r_1) \\
& c_2 \leftarrow E(P_{k2}, m; r_2) \\
& \text{let } x = (c_1, c_2) \text{ in } // \text{ statement to prove} \\
& \text{let } w = (r_1, r_2, m) \text{ in } // \text{ witness for the statement} \\
& \pi \leftarrow P(\sigma, x, w) \\
& \text{let } c = (c_1, c_2, \pi) \text{ in} \\
& \text{return } c
\end{aligned}
$$

$$
\begin{aligned}
D'(S'_k, c): \quad & \text{let } (S_{k1}) = S'_k \text{ in} \\
& \text{let } (c_1, c_2, \pi) = c \text{ in} \\
& \text{let } x = (c_1, c_2) \text{ in} \\
& \text{if } V(\sigma, x, \pi) \\
& \quad \text{then return } D(S_{k1}, c_1) \\
& \quad \text{else return } \perp
\end{aligned}
$$

The correctness of (G', E', D') is easy. If the keys were generated correctly and the messages were encrypted correctly, then π is a valid proof for the fact that c_1 and c_2 encrypt the same message. So the verifier $V(\sigma, x, \pi)$ will output true and the original message m will be obtained by the decryption $D(S_{k1}, c_1)$.
Now we want to prove that (G', E', D') is IND-CCA1-secure. Let Sim be a simulator of the NIZK proof system.
Consider the games 0, 1, 2, 2', 3, 3', 4 shown in Table 4.
In these games, the adversary is given a decrypting oracle during the first phase and at the end the adversary should output a bit to distinguish which game he is playing. We want to show that the adversary A cannot distinguish between Game 0 and Game 4.
We do this by a hybrid argument, showing that A cannot distinguish between any two consecutive games.
Game 1 differs from Game 0 by the use of a simulator for the random string and proof generation. They are indistinguishable by reduction to the zero-knowledge property of the proof system.

Game 2 differs from Game 1 in that it obtains ciphertext c_2 from the message m_1. They are indistinguishable by reduction to the IND-CPA property of the underlying PKE, which guarantees that encryptions of m_0 cannot be distinguished from encryptions of m_1.
Game 2' differs from Game 2 by using the key S_{k2} for decryption instead of S_{k1}. The adversary's view of the two games only differs if the event FAKE occurs. FAKE is the event that A submits a query (c_1, c_2, π) for its decryption oracle such that $D(S_{k1}, c_1) \neq D(S_{k2}, c_2)$ but $V(\sigma, (c_1, c_2), \pi) = 1$. This happens with a negligible probability.
Game 3 differs from Game 2' in that it obtains ciphertext c_1 from the message m_1. They are indistinguishable by reduction to the IND-CPA property of the underlying PKE, which guarantees that encryptions of m_0 cannot be distinguished from encryptions of m_1.
Game 3' differs from Game 3 by using the key S_{k1} for decryption instead of S_{k2}. The adversary's view of the two games only differs if the event FAKE occurs, which happens with a negligible probability.
Game 4 differs from Game 3' by the use of the real NIZK proof system instead of a simulator. They are indistinguishable by reduction to the zero-knowledge property of the proof system.
More details on the proof are given in the next lecture. ■

Game 0	Game 1
$(P_{k1}, S_{k1}), (P_{k2}, S_{k2}) \leftarrow G(1^k)$ $\sigma \leftarrow K(1^k)$ let $P'_k = (P_{k1}, P_{k2}, \sigma)$ and $S'_k = (S_{k1})$ in $(m_0, m_1) \leftarrow A^{D'(S'_k,\cdot)}(P'_k)$ $c_1 \leftarrow E(P_{k1}, m_0; r_1), c_2 \leftarrow E(P_{k2}, m_0; r_2)$ $\pi \leftarrow P(\sigma, (c_1, c_2), (r_1, r_2, m_0))$ $b \leftarrow A(P'_k, c_1, c_2, \pi)$	$(P_{k1}, S_{k1}), (P_{k2}, S_{k2}) \leftarrow G(1^k)$ $\sigma \leftarrow Sim(1^k)$ let $P'_k = (P_{k1}, P_{k2}, \sigma)$ and $S'_k = (S_{k1})$ in $(m_0, m_1) \leftarrow A^{D'(S'_k,\cdot)}(P'_k)$ $c_1 \leftarrow E(P_{k1}, m_0; r_1), c_2 \leftarrow E(P_{k2}, m_0; r_2)$ $\pi \leftarrow Sim(c_1, c_2)$ $b \leftarrow A(P'_k, c_1, c_2, \pi)$
Game 2	Game 2'
$(P_{k1}, S_{k1}), (P_{k2}, S_{k2}) \leftarrow G(1^k)$ $\sigma \leftarrow Sim(1^k)$ let $P'_k = (P_{k1}, P_{k2}, \sigma)$ and $S'_k = (S_{k1})$ in $(m_0, m_1) \leftarrow A^{D'(S'_k,\cdot)}(P'_k)$ $c_1 \leftarrow E(P_{k1}, m_0; r_1), c_2 \leftarrow E(P_{k2}, m_1; r_2)$ $\pi \leftarrow Sim(c_1, c_2)$ $b \leftarrow A(P'_k, c_1, c_2, \pi)$	$(P_{k1}, S_{k1}), (P_{k2}, S_{k2}) \leftarrow G(1^k)$ $\sigma \leftarrow Sim(1^k)$ let $P'_k = (P_{k1}, P_{k2}, \sigma)$ and $S'_k = (S_{k2})$ in $(m_0, m_1) \leftarrow A^{D'(S'_k,\cdot)}(P'_k)$ $c_1 \leftarrow E(P_{k1}, m_0; r_1), c_2 \leftarrow E(P_{k2}, m_1; r_2)$ $\pi \leftarrow Sim(c_1, c_2)$ $b \leftarrow A(P'_k, c_1, c_2, \pi)$
Game 3	Game 3'
$(P_{k1}, S_{k1}), (P_{k2}, S_{k2}) \leftarrow G(1^k)$ $\sigma \leftarrow Sim(1^k)$ let $P'_k = (P_{k1}, P_{k2}, \sigma)$ and $S'_k = (S_{k2})$ in $(m_0, m_1) \leftarrow A^{D'(S'_k,\cdot)}(P'_k)$ $c_1 \leftarrow E(P_{k1}, m_1; r_1), c_2 \leftarrow E(P_{k2}, m_1; r_2)$ $\pi \leftarrow Sim(c_1, c_2)$ $b \leftarrow A(P'_k, c_1, c_2, \pi)$	$(P_{k1}, S_{k1}), (P_{k2}, S_{k2}) \leftarrow G(1^k)$ $\sigma \leftarrow Sim(1^k)$ let $P'_k = (P_{k1}, P_{k2}, \sigma)$ and $S'_k = (S_{k1})$ in $(m_0, m_1) \leftarrow A^{D'(S'_k,\cdot)}(P'_k)$ $c_1 \leftarrow E(P_{k1}, m_1; r_1), c_2 \leftarrow E(P_{k2}, m_1; r_2)$ $\pi \leftarrow Sim(c_1, c_2)$ $b \leftarrow A(P'_k, c_1, c_2, \pi)$
Game 4	
$(P_{k1}, S_{k1}), (P_{k2}, S_{k2}) \leftarrow G(1^k)$ $\sigma \leftarrow K(1^k)$ let $P'_k = (P_{k1}, P_{k2}, \sigma)$ and $S'_k = (S_{k1})$ in $(m_0, m_1) \leftarrow A^{D'(S'_k,\cdot)}(P'_k)$ $c_1 \leftarrow E(P_{k1}, m_1; r_1), c_2 \leftarrow E(P_{k2}, m_1; r_2)$ $\pi \leftarrow P(\sigma, (c_1, c_2), (r_1, r_2, m_1))$ $b \leftarrow A(P'_k, c_1, c_2, \pi)$	

Table 4: Games used for the hybrid argument.