0368.4162 Foundations of Cryptography Fall 2017	Nir Bitansky

Final Exam, Moed A, Solution

February 4 2018

1. Let (E, D) be a 1-KPA-secure secret-key encryption for messages of length $n + 1$ (for key length n).

   (a) Assume that the encryption algorithm E is deterministic. Prove that the following function family $f = \{f_n\}_{n \in \mathbb{N}}$ is one-way or give a counter example:

   $$\forall sk \in \{0,1\}^n : f_n(sk) = E_{sk}(0^{n+1}) \ .$$

   Solution: We'll show that if there is an efficient(w.l.o.g deterministic) adversary A that inverts f with probability ε, then A can be used to distinguish $E_{sk}(0^{n+1})$ from $E_{sk}(U_{n+1})$ with advantage $\varepsilon/2$. Indeed, the number of ciphertexts ct that A manages to invert is at most $\varepsilon 2^n$. On the other hand, $E_{sk}(\cdot)$ is injective, implying that $E_{sk}(U_{n+1})$ is uniformly distributed over a set of ciphertexts of size 2^{n+1}, and is thus inverted with probability at most $\varepsilon/2$. This gives rise to the required distinguisher — given ct it tries to invert using A, and outputs 1 if and only if A succeeds.

   (b) Assume that the encryption algorithm E also uses randomness r of some polynomial length $\ell(n)$. Prove that the following function family $f = \{f_n\}_{n \in \mathbb{N}}$ is one-way or give a counter example:

   $$\forall (sk, r) \in \{0,1\}^n \times \{0,1\}^{\ell(n)} : f_n(sk, r) = E_{sk}(0^{n+1}; r) \ .$$

   Solution: We'll give a counter example. Let $G : \{0,1\}^n \to \{0,1\}^{n+1}$ be a PRG (recall that an encryption scheme for messages of length $> n$ implies OWFs and thus also PRGs). Define

   $$\forall sk \in \{0,1\}^n, r, m \in \{0,1\}^{n+1} :$$
   $$E_{sk}(m; r) = \begin{cases} (G(sk) \oplus m, 0^{n+1}) & \text{if } sk \neq 1^n \\ (r, m) & \text{if } sk = 1^n \end{cases}$$
   $$D_{sk}(c_1, c_2) = \begin{cases} G(sk) \oplus c_1 & \text{if } sk \neq 1^n \\ c_2 & \text{if } sk = 1^n \end{cases} \ .$$

   Correctness follows readily. By the pseudorandomness of G, and the fact that $sk = 1^n$ w.p. at most 2^{-n}, it holds that $E_{sk}(m; r) \approx_c (U_{n+1}, 0^{n+1})$, and thus the scheme is 1-KPA secure. However, we can invert the corresponding OWF, with probability 1. Given an image $f_n(sk, r) = (c_1, 0^n)$, we return the preimage $(1^n, c_1)$.

2. Let (G, E, D) be a CPA-secure public-key encryption scheme that is (perfectly) correct. For each of the following suggestions, prove that it is a (perfectly) binding and computationally hiding commitment scheme, or give a counter example.

   (a)
   $$Com(m; (r_g, r_e)) = (pk, E_{pk}(m; r_e)) \ ,$$

   where m is the committed message, (r_g, r_e) are the randomness used by the commitment, each sampled at random and independently from $\{0,1\}^n$, pk is generated by $G(1^n; r_g)$, with random coins r_g, and r_e is the randomness used by the encryption algorithm.

Solution: We'll prove that the scheme is a commitment. The hiding of the commitment follows directly from CPA security — for any two messages m, m':

$$Com(m) = pk, E_{pk}(m) \approx_c pk, E_{pk}(m') \approx_c Com(m') \ .$$

To see that binding holds, note that if $Com(m,(r_g,r_e)) = Com(m',(r'_g,r'_e))$, then for $(sk,pk) = G(1^n;r_g)$ and $(pk'sk') = G(1^n;r'_g)$, it holds that $pk = pk'$ and

$$m = D_{sk}(E_{pk}(m;r_e)) = D_{sk}(E_{pk}(m';r'_e)) = m' \ .$$

(b)

$$Com(m;(r_g,r_e)) = E_{pk}(m;r_e) \ ,$$

where all parameters are generated as in the previous item.

Solution: We'll construct a counter example. Specifically, given any public-key encryption scheme (G',E',D'), we'll construct a new bit-encryption scheme (G,E,D) such that the above is not binding. Let us say that a secret/public key k is consistent with randomness $r_g \in \{0,1\}^n$, if $G(1^n;r_g)$ outputs k as the secret/public key (note that we can efficiently check if a given key k is consistent with given randomness r_g). Assume w.l.o.g that in (G',E',D'), no key k is consistent with both 0^n and 1^n.

In our new scheme:

- G is the same as G'.
- $E_{pk}(m)$: if pk is consistent with randomness 0^n, resample $(sk',pk') = G(1^n;r'_g)$ for randomness, $r'_g = 1^n$. Output $ct = E'_{pk'}(m \oplus 1)$.
- $D_{sk}(ct)$: if sk is consistent with randomness 0^n, resample $(sk',pk') = G(1^n;r'_g)$ for randomness, $r'_g = 1^n$. Output $1 \oplus D'_{sk'}(ct)$.

The new scheme is CPA secure, as we've only changed it on negligible fraction of keys. It is also still perfectly correct — we changed it only on keys consistent for with 0^n, where we shifted to using keys consistent with 1^n, and consistently flipped/unflipped the encrypted bit during encryption/decryption.

Now, however, we have that for any r_e,

$$Com(0;(0^n,r_e)) = Com(1;(1^n,r_e)) \ .$$

3. A triangle in a graph consists of three vertices that are all connected to each other by edges. Consider a variant of the GMW zero-knowledge proof system for 3COL where (after the prover commits to a coloring) instead of requesting that the prover opens a random edge, the verifier first flips a random coin $b \leftarrow \{0,1\}$: if $b = 0$, or there are no triangles in the graph, the verifier asks that the prover opens a random edge as in the original protocol, whereas if $b = 1$, and there are triangles, the verifier asks that the prover opens a random triangle. As in the original protocol, the verifier accepts if for every edge that the prover opened, the colors revealed are distinct.

   (a) Is the protocol still zero-knowledge. If your answer is no, give a counter example. If your answer is yes, describe a simulator (no need to prove validity).

   Solution: The protocol is still zero-knowledge. Assume w.l.o.g the graph does have triangles (otherwise, the protocol is the same as the original GMW protocol, and simulation is done in the same way). The simulator first guesses $b' \leftarrow \{0,1\}$. If $b' = 0$, the simulator proceeds as in the original GMW simulation — it guesses $e' = (u,v) \leftarrow E$, chooses random distinct colors for u and v and then gives the verifier a commitment to these colors for u and v as well as to arbitrary colors for the rest of the vertices. If $b' = 1$, the simulator chooses a random triangle $t' = (u, v, w) \leftarrow T$ from the set of all triangles T in G, and chooses three random distinct colors for u, v, w. Again it gives the verifier a commitment to theses colors for u, v and w, and to arbitrary colors for the rest of the vertices. Then, when the verifier presents its choice b and edge e or triangle t, if they are inconsistent with the simulators guess b' and e' or t', the simulator goes back to the first step of guessing. Otherwise, it opens the required commitments.

(b) Consider $t = 20|E|$ sequential repetitions of the above protocol. Show that there exists an efficient extractor algorithm E such that given every graph $G = (U, E)$ and the code of a deterministic prover P^* that with probability $1/100$ convinces the verifier V of accepting G, the extractor outputs a valid 3-coloring of G with probability 0.99. The extractor's running time should be polynomial in $|G|$ and the worst-case running time t of the prover P^*.

Solution: Similarly to what we've seen in the homework, with probability at least

$$\frac{1}{100} - \left(1 - \frac{1}{2|E|}\right)^t > 1/200 \ ,$$

in a random interaction with the prover P^*, there will exist a session $i \in [t]$ where the prover convinces the verifier with probability greater than $\left(1 - \frac{1}{2|E|}\right)$.

This means that in this session, for any verifier choice $b = 0, e \in E$, the prover will reveal a valid coloring. Our extractor will attempt to extract a coloring from such a session. It will sample t sequential sessions, and then attempt to extract from each one of them, by rewinding the prover, and asking it to reveal for every choice $b = 0, e \in E$. This succeeds with probability at least $1/200$, and can be amplified to 0.99, by independently repeating a sufficiently large constant number of times.