University of Maryland
CMSC858K — Introduction to Cryptography
Professor Jonathan Katz

Problem Set 5

1. I show here how to obtain a scheme with complexity $n^{1/3}$; the extension to n^ϵ applies the same idea recursively.

   It will help to first recall the basic scheme. Let the database have size d, and use n as the length of the modulus. The server views the database as an $d^{1/2}$-by-$d^{1/2}$ table; the user sends $d^{1/2}$ elements of $\mathbb{Z}_N^*$, of which exactly one is a quadratic residue. The server then sends back $d^{1/2}$ elements of $\mathbb{Z}_N^*$ but the user only looks at one of them to determine the bit of the database that he is interested in. The total communication complexity is $2nd^{1/2}$.

   Since the user only looks at one of the elements returned by the server, a natural idea is to use PIR recursively to retrieve the element he is interested in. By itself, this will not improve the communication complexity since the user is already sending $n \cdot d^{1/2}$ bits to the server.

   What we do instead is the following. View the database as a $d^{1/3}$-by-$d^{2/3}$ array. The user now sends $d^{1/3}$ elements of $\mathbb{Z}_N^*$, where again only one of these (corresponding to the row of the bit he is interested in) is a quadratic residue. The server prepares $d^{2/3}$ elements of $\mathbb{Z}_N^*$ for the response *but does not send them.* As before, the user is only interested in one of these elements. So the user and the server can now apply the basic scheme to these $d^{2/3}$ elements. The communication complexity of this step is $n \cdot 2n(d^{2/3})^{1/2} = 2n^2 \cdot d^{1/3}$ (the extra factor of n arises because the user wants n bits), for a total communication complexity of $(2n^2+2n) \cdot d^{1/3}$.

   Note that although the above process was described as taking two stages, it can in fact all be done using one message from the user to the server and one message in response.

2. (a) By construction, the user's index I always appears in exactly one of the queries sent to the databases. So the probability here is 1.

   (b) Fixing i, the query (S_i, T_i, U_i) received by server i is just three uniformly random sets. So the probability that I is in the query is 1/8.

   Alternately, one could argue that since exactly one of the eight servers receives a query containing I, by symmetry the probability that server i receives such a query is 1/8.

   (c) Following the first reasoning above, the probability is still 1/8.

   (d) Since the answer to both (b) and (c) was the same, the probability the database correctly guesses the user's index is exactly 1/2 (or no better than random guessing).

3. Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a one-time signature scheme for 1-bit messages. Assume without much loss of generality that on input 1^n, algorithm Gen uses n random bits. (You can either prove that this is without loss of generality, or simply adapt the following argument to the general case.) Define f as follows: on input $r \in \{0,1\}^n$, compute $\mathsf{Gen}(1^n; r)$. Note that this is a *deterministic* computation even though Gen is a randomized algorithm, since we use r as the random coins of Gen. It is not very difficult to show that f as defined must be one-way (assuming, as usual in this class, perfect correctness of the signature scheme).

4. There are a number of ways to approach this problem. I show here how to request a signature on a single message and then forge a signature on another message. Note that $0\|m\|0^{\ell/10} = m \cdot 2^{\ell/10} \bmod N$. Request a signature on $m = 0\cdots011$ (which is '3' in binary) and obtain signature σ. The claim is that $\sigma^2 \bmod N$ is a forgery on the message $m' = 0\cdots0\,1001\,0^{\ell/10}$ (which is $9 \cdot 2^{\ell/10}$ in binary). Indeed,

$$\begin{aligned}
(\sigma^2)^e &= (\sigma^e)^2 \bmod N \\
&= (0\|m\|0^{\ell/10})^2 \bmod N \\
&= (3 \cdot 2^{\ell/10})^2 \bmod N \\
&= 9 \cdot 2^{\ell/10} \cdot 2^{\ell/10} \bmod N \\
&= 0\|0\cdots0\,1001\,0^{\ell/10}\|0^{\ell/10} \bmod N.
\end{aligned}$$

5. I give the solution in each case, but leave the proof to the reader.

   (a) Let f be a one-way function, and define f' as follows: $f'(x_1\cdots x_n) = f(x_1\cdots x_{n-1}0)$. It is not hard to show that f' is still one-way, but plugging f' into Lamport's scheme gives a scheme that is *not* strongly secure.

   (b) Let H be a collision-resistant hash function mapping strings of length $2n$ to strings of length n. It is not hard to show that H is one-way. Using H in Lamport's scheme gives a strongly-secure one-time signature scheme.

   (Note: formally we need to speak in terms of keyed hash functions but it is not hard to modify Lamport's scheme to take this into account.)