MTAT.07.003 Cryptology II
Spring 2012 / Exercise session ?? / Example Solution

Exercise (Electronic lottery). *A company holds electronic lotteries briefly described below. A secure hardware is used to generate the output* $x \in \{0,1\}^{20}$. *The input* x *is fed into a specially designed garbling algorithm (worldwide patented), which takes in* x *and broadcasts a garbled output* y *to everybody. All participants submit their guesses* q_i *about* x. *The company releases* x *and everybody uses the description of the garbling algorithm to verify that* y *was computed form* x. *The one who guessed* x *gets a prize.*

1. *Formalize the lottery system using abstract primitives for appropriate actions. Describe such functional requirements that the organizer cannot cheating and that participants can verify the correctness.*
2. *Define an attack scenario where participants try to cheat. Quantify the success of the malicious participant. Formalize the corresponding security definition.*
3. *Show that no garbling algorithm can meet functional requirements and be secure at the same time.*

Solution. The lottery system can be viewed as a public function $f : \{0,1\}^{20} \to \mathcal{Y}$ together with a broadcast mechanism that provides authentic transmission of y value to all participants of the lottery.

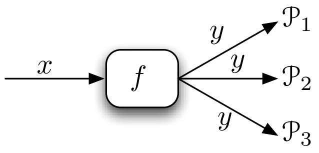

Figure 1: Lottery design for three participants

The easiest way to assure that the organiser of the lottery cannot cheat is to assure that f is one-to-one mapping, i.e., there are no 20-bit strings $x_1 \neq x_2$ that would hash to the same value y. Then everybody can verify the legitimacy of the lottery by computing $f(x)$ and comparing it with y. A more careful analysis shows that f does not have to be one-to-one mapping as long as the organiser cannot easily find two different inputs that hash to the same value. However, this condition cannot be satisfied unless the function is injective due to the small domain size. As there are only 2^{20} possible inputs, the organiser can evaluate f on each of them and quickly find and tabulate all possible collisions $f(x_1) = f(x_2)$.

Let us now consider security against malicious participants of the lottery. For clarity, assume that organises of the lottery collect guesses during a single day, i.e., each participant has $86,400$ seconds to submit their guess. Given that an ordinary computer can do 10^9 operation per second, an attacker must construct an algorithm that gives an output after 10^{14} elementary operations. Of course, this is a rough lower bound on available computing power. For example, the adversary can rent large computing cluster or use botnet to crack the function f. Although these actions may increase computational power of an adversary, we can always fix some (not so) well-justified limit t of computational operations, which is infeasible for all potential adversaries or has to high cost compared to final gains. Hence, the protocol is secure if all t-time algorithms $\mathcal{A}$ does not have overly big probability to win the following game

$$\mathcal{G}^{\mathcal{A}}\left[\begin{array}{l} x \xleftarrow{u} \{0,1\}^{20} \\ x_* \leftarrow \mathcal{A}(f(x)) \\ \textbf{return } [x_* \overset{?}{=} x] \end{array}\right.$$

which captures the honest lottery organisation. There are several aspects to note here. First, we do not have to give f as an input to the adversary $\mathcal{A}$. The function f is fixed and we can always assume that its description is directly hardwired into the code of $\mathcal{A}$. Second, one can easily win the lottery with probability 2^{-20} by just choosing a single value x_*. Hence, it is common to measure the additional gain $\mathcal{A}$ gets from seeing $f(x)$:

$$\mathsf{Adv}(\mathcal{A}) = \Pr\left[\mathcal{G}^{\mathcal{A}} = 1\right] - 2^{-20} \ ,$$

where $\mathcal{G}^{\mathcal{A}}$ denotes the output of the game and the probability is taken over the choice of x and random coins used by $\mathcal{A}$. To be concrete, lets postulate that the additional gain is negligible if it is below 10^{-6}. You will be murdered in a year with this probability and yet you do not worry about it. Also, the probability that a legitimate user wins the lottery with probability about 10^{-6}.

Now if we assume that f is one-to-one mapping, it is rather straightforward to invert f. Given input y, we can try all possible inputs $x \in \{0,1\}^{20}$ and outputs the value if $f(x) = y$. Hence, after 2^{20} evaluations of f, the adversary $\mathcal{A}$ wins the game $\mathcal{G}$ with maximal probability 1. Moreover, if one tries values of x in random order, then success after k evaluations is approximately $k \cdot 10^{-6}$ and thus advantage around 10^{-6} is achievable only if one can evaluate f on a single value during a day and even then the adversary can significantly increase its chances by cloud computing.