MTAT.07.003 Cryptology II
Spring 2010 / Exercise session I

1. 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.

   (a) Formalise the lottery system using abstract primitives for appropriate actions. Describe such functional requirements that the organiser cannot cheating and that participants can verify the correctness.

   (b) Define an attack scenario where participants try to cheat. Quantify the success of the malicious participant. Formalise the corresponding security definition.

   (c) Show that no garbling algorithm can meet functional requirements and be secure at the same time.

2. A standard way to protect data against malicious corruption is hashing. Namely, there are many industry standard algorithms, like MD5, SHA-256 and WHIRLPOOL, that take in a long file and output a short digest. If the digest is securely stored, then the validity of the file can be tested by recomputing the digest and comparing it to the stored value.

   (a) Formalise the functional requirements and describe the attack scenario if the original data is generated by flipping a fair coin.

   (b) Describe the attack scenario if an attacker gets to know the randomly generated original data before spoofing the file.

   (c) Describe the attack scenario if attacker can influence the content of the original file. Show that no function can be secure against such attacks. What does it mean in real life applications?

3. Salted hashing is common mechanism for storing passwords. Each user is first granted an identifier `id`. Every time the password is hashed, the identifier `id` is appended to it and then a system-wide hash function is used to compute the digest. The authentication is successful if the digest coincides with the digest stored in `passwd` file.

   (a) Formalise the system by using abstract primitives for appropriate actions. Describe the functional requirements that are needed for seamless authentication. Compare the formalisation with the lottery system described in the first exercise.

(b) Define an attack scenario where the attacker tries to reverse engineer passwords from salted hashes. Formalise the corresponding security condition. Can this security condition be met in practice?

(c) Extend the attack description to the setting where the identifier `id` depends on the identity of a user. What does such a design choice give to the attacker? Modify the corresponding attack scenario and derive the corresponding security requirement.

4. Let $\mathbb{G}$ be a finite group such that all elements $y \in \mathbb{G}$ can be expressed as powers of $g \in \mathbb{G}$. Then the Computational Diffie-Hellman (CDH) problem is following. Given $x = g^a$ and $y = g^b$, find a group element $z = g^{ab}$.

   (a) Show that Computational Diffie-Hellman problem is random self-reducible, i.e., for any algorithm $\mathcal{B}$ that achieves advantage

   $$\mathsf{Adv}^{\mathsf{cdh}}_{\mathbb{G}}(\mathcal{B}) \doteq \Pr\left[x, y \xleftarrow[u]{} \mathbb{G} : \mathcal{B}(x, y) = g^{\log_g x \log_g y}\right]$$

   there exists an oracle algorithm $\mathcal{A}^{\mathcal{B}}$ that for any input $x, y \in \mathbb{G}$ outputs the correct answer with the probability $\mathsf{Adv}^{\mathsf{cdh}}_{\mathbb{G}}(\mathcal{B})$ and has roughly the same running time.

   (b) Given that the CDH problem is random self-reducible, show that the difficulty of CDH instances cannot wary a lot. Namely, let $\mathcal{B}$ be a $t$-time algorithm that achieves maximal advantage $\mathsf{Adv}^{\mathsf{cdh}}_{\mathbb{G}}(\mathcal{B})$. What can we say about worst-case advantage

   $$\min_{x,y} \Pr\left[\mathcal{A}(x, y) = g^{\log_g x \log_g y}\right]?$$

   Can there be a large number of pairs $(x, y)$ for which the CDH problem is easy?

5. Let $\mathbb{G}$ be a finite group such that all elements $y \in \mathbb{G}$ can be expressed as powers of $g \in \mathbb{G}$. Then the Decisional Diffie-Hellman (DDH) problem is following. Given $x = g^a$ and $y = g^b$ and $z$, decide whether $z = g^{xy}$ or not.

   (a) Show that Decisional Diffie-Hellman problem can be reduced to Computational Diffie-Hellman problem, i.e., for any algorithm $\mathcal{A}$ that achieves advantage $\mathsf{Adv}^{\mathsf{cdh}}_{\mathbb{G}}(\mathcal{A})$, there exists an oracle algorithm $\mathcal{B}^{\mathcal{A}}$ that has has roughly the same running time and that the advantage

   $$\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{B}) \doteq \left|\Pr\begin{bmatrix} x, y, z \xleftarrow[u]{} \mathbb{G} : \\ \mathcal{B}(x, y, z) = 1 \end{bmatrix} - \Pr\begin{bmatrix} x, y \xleftarrow[u]{} \mathbb{G}, z \leftarrow g^{\log x \log y} : \\ \mathcal{B}(x, y, z) = 1 \end{bmatrix}\right|$$

   is roughly equal to the advantage $\mathsf{Adv}^{\mathsf{cdh}}_{\mathbb{G}}(\mathcal{A})$.

   (b) Provide a reductions between DL, CDH and DDH problems.

   (c) Show that if there exists an efficient procedure that can always compute the highest bit of $\log_g y$ then the DL problem is easy.

($\star$) Although the Decisional Diffie-Hellman problem is randomly self-reducible, naively re-randomised challenges are dependent on each other. Is it possible to construct an algorithm that amplifies the advantage by running several instances of the base algorithm to provide an aggregated answer. Give a construction or a proof that such a construction cannot exist.