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

Exercise (Random self-reducibility of DDH). *Let $\mathbb{G} = \langle g \rangle$ be a finite group of a prime order q generated by the powers of an element g. Then the Decisional Diffie-Hellman (DDH) problem is following. For any triple $x, y, z \in \mathbb{G}$, you must decide whether it is a Diffie-Hellman triple or not. Formally, the corresponding distinguishing task is specified through two games:*

$$
\begin{array}{ll}
\mathcal{Q}_0^{\mathcal{B}} & \qquad\qquad \mathcal{Q}_1^{\mathcal{B}} \\
\left[\begin{array}{l} a, b \xleftarrow{u} \mathbb{Z}_q \\ c \xleftarrow{u} \mathbb{Z}_q \\ \textbf{\textit{return}}\ \mathcal{B}(g^a, g^b, g^c) \end{array}\right. & \qquad\qquad \left[\begin{array}{l} a, b \xleftarrow{u} \mathbb{Z}_q \\ c \leftarrow ab \\ \textbf{\textit{return}}\ \mathcal{B}(g^a, g^b, g^c) \end{array}\right.
\end{array}
$$

where the advantage is computed as $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{A}) = |\Pr[\mathcal{G}_0^{\mathcal{A}} = 1] - \Pr[\mathcal{G}_1^{\mathcal{A}} = 1]|$. Show that DDH problem is random self-reducible and sketch how to amplify the success probability by majority voting.

Solution. For a solution, we first discuss what random self-reducibility means in the context of Decisional Diffie-Hellman problem. Then we show how to achieve random self-reducibility and what are the consequences. As a last step, sketch how to amplify the success probability by majority voting.

Definition of Random Self-reducibility. It is easy to formalise random self-reducibility for a discrete logarithm or Computational Diffie-Hellman problem, as the problem is formalized through a single game. For a Decisional Diffie-Hellman problem, we have two security games $\mathcal{Q}_0$ and $\mathcal{Q}_1$. Still, we could require existence of an algorithm $\mathcal{A}^{\mathcal{B}}$ such that for any challenge tuple $g^{a_0}, g^{b_0}, g^{c_0}$ generated in the $\mathcal{G}_0$ and a challenge tuple $g^{a_1}, g^{b_1}, g^{c_1}$ generated in the $\mathcal{G}_1$, the corresponding advantage

$$\mathsf{Adv}(\mathcal{A}) = |\Pr[\mathcal{A}(g^{a_0}, g^{b_0}, g^{c_0}) = 1] - \Pr[\mathcal{A}(g^{a_1}, g^{b_1}, g^{c_1}) = 1]| = \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{B}) \ .$$

However, such a goal is clearly unachievable since a valid Diffie-Hellman tuple $g^{a_1}, g^{b_1}, g^{c_1}$ can is also generated in the game $\mathcal{G}_0$, as well. Consequently, more details are needed in the formalisation.

Note that all triples $\mathbb{G} \times \mathbb{G} \times \mathbb{G}$ can be divided into Diffie-Hellman triples and non-Diffie-Hellman triples. To show random self-reducibility for a decision problem, we must provide a re-randomisation algorithm $\mathcal{R}$, which takes any DH triple to a random DH triple and any non-DH triple to a random non-DH triple. Then it is straightforward to construct $\mathcal{A}$ as $\mathcal{B}(\mathcal{R}(x, y, z))$ such that

$$\mathsf{Adv}(\mathcal{A}) \approx \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{B})$$

for any $(g^{a_0}, g^{b_0}, g^{c_0})$ non-DH tuple and $(g^{a_1}, g^{b_1}, g^{c_1})$ DH tuple pair. However note that we do not obtain the precise equality as $\mathcal{B}$ sees only random non-DH tuples and not random group elements as in the game $\mathcal{Q}_0$. To fix this cosmetic issue we might require that the re-randomisation algorithm $\mathcal{R}$, must take any DH triple to a random DH triple and any non-DH triple to a random triple. Then

$$\mathsf{Adv}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{B})$$

for any $(g^{a_0}, g^{b_0}, g^{c_0})$ non-DH tuple and $(g^{a_1}, g^{b_1}, g^{c_1})$ DH tuple pair as desired.

Construction for random self-reducibility. Let $\mathbb{G}$ be a q-element group and (x, y, z) either a DH or non-DH tuple. Then the following re-randomisation algorithm

$$
\begin{array}{l}
\mathcal{R}(x, y, z) \\
\left[\begin{array}{l} u, v, w \xleftarrow{u} \mathbb{Z}_q \\ x_* \leftarrow x^w g^u \\ y_* \leftarrow y g^v \\ z_* \leftarrow z^w x^{vw} y^u g^{uv} \\ \textbf{return}\ (x_*, y_*, z_*) \end{array}\right.
\end{array}
$$

takes DH triple to a random DH triple and non-DH triple to a random triple.

Analysis. By looking discrete logarithms we can simplify further analysis. Let us use shorthands

$$\begin{aligned} a &= \log(x) & a_* &= \log(x_*) \\ b &= \log(y) & b_* &= \log(y_*) \\ c &= \log(z) & c_* &= \log(z_*) \end{aligned}$$

for denoting discrete logarithms for the inputs and outputs. Then (x, y, z) is a DH triple iff $ab = c$ and (x_*, y_*, z_*) is a DH triple iff $a_* b_* = c_*$. By the construction of re-randomiser

$$\begin{aligned} a_* &= aw + u \\ b_* &= b + v \\ c_* &= cw + awv + bu + uv \ , \end{aligned}$$

and thus

$$a_* b_* - c_* = (aw + u)(b + v) - cw - awv - bu - uv = (ab - c)w \ .$$

Consequently, the re-randomisation algorithm returns a DH triple whenever the input (x, y, z) is a DH triple. As a_* and b_* are independent and uniformly distributed over $\mathbb{Z}_q$, re-randomisation returns all DH tuples with uniform probability. If (x, y, z) is non-DH triple, then $ab \neq c$ then the output can be any triple. Indeed, the system of linear equations

$$\begin{aligned} a_* &= aw + u \\ b_* &= b + v \\ c_* &= cw + awv + bu + uv \end{aligned} \tag{1}$$

can be solved for any target (a_*, b_*, c_*) by taking

$$\begin{aligned} v &= b_* - b \\ w &= \frac{a_* b_* - c_*}{ab - c} \\ u &= \frac{a_*(ab - c) - a(a_* b_* - c_*)}{ab - c} \ . \end{aligned}$$

As each of those combinations (u, v, w) have equal probability by the construction of $\mathcal{R}$, the distribution of (x_*, y_*, z_*) must be uniform over $\mathbb{G} \times \mathbb{G} \times \mathbb{G}$.

Smoothed distinguisher. Given a re-randomiser $\mathcal{R}$ and a distinguisher $\mathcal{B}$ we can construct a distinguisher

$$\begin{aligned} &\mathcal{A}(x, y, z) \\ &\left[\begin{array}{l} (x_*, y_*, z_*) \leftarrow \mathcal{R}(x, y, z) \\ \textbf{return } \mathcal{B}(x_*, y_*, z_*) \end{array}\right. \end{aligned}$$

that works equally well for all DH-tuple vs non-DH tuple pairs. Indeed, for a DH tuple (x, y, z) the new challenge (x_*, y_*, z_*) is uniformly chosen DH-tuple and thus the program is equivalent to

$$\begin{aligned} &\mathcal{Q}_1^{\mathcal{B}} \\ &\left[\begin{array}{l} a_*, b_* \xleftarrow[u]{} \mathbb{Z}_q \\ c_* \leftarrow ab \\ \textbf{return } \mathcal{B}(g^{a_*}, g^{b_*}, g^{c_*}) \ . \end{array}\right. \end{aligned}$$

For a non-DH tuple the new challenge (x_*, y_*, z_*) is uniformly chosen over $\mathbb{G} \times \mathbb{G} \times \mathbb{G}$ and thus the program is equivalent to

$$
\begin{aligned}
&\mathcal{Q}_0^{\mathcal{B}} \\
&\left[\begin{array}{l}
a_*, b_* \xleftarrow[u]{} \mathbb{Z}_q \\
c_* \xleftarrow[u]{} \mathbb{Z}_q \\
\textbf{return } \mathcal{B}(g^{a_*}, g^{b_*}, g^{c_*})
\end{array}\right. .
\end{aligned}
$$

Consequently, we have obtained the desired bound $\mathsf{Adv}(\mathcal{A}) = \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(\mathcal{B})$.

Alternative construction for random self-reducibility. Note that the re-randomisation algorithm $\mathcal{R}$ completely ignores the input z whenever $w = 0$ and fabricates a DH-tuple even if the input is not a DH tuple. By correcting this error we obtain a new re-randomisation algorithm

$$
\begin{aligned}
&\mathcal{R}^*(x, y, z) \\
&\left[\begin{array}{l}
w \xleftarrow[u]{} \mathbb{Z}_q^* \\
u, v \xleftarrow[u]{} \mathbb{Z}_q \\
x_* \leftarrow x^w g^u \\
y_* \leftarrow y g^v \\
z_* \leftarrow z^w x^{vw} y^u g^{uv} \\
\textbf{return } (x_*, y_*, z_*)
\end{array}\right.
\end{aligned}
$$

that still takes DH triple to a random DH triple and non-DH triple to a random non-DH triple. The previous argumentation remains intact. In particular, the analysis if the input is a DH-tuple is identical. For non-DH tuples, most of the results still hold. In particular, $a_* b_* - c_* = (ab - c)w$ is nonzero for all $w \in \mathbb{Z}_q^*$ and thus (x_*, y_*, z_*) cannot be a DH triple. The output must have uniform distribution over non-DH triples, as the solution (u, v, w) to system of linear equations (1) satisfies the constraint $w \neq 0$ for all non-DH tuples.

Smoothed distinguisher. By combining the re-randomiser $\mathcal{R}^*$ and distinguisher, we obtain a distinguisher

$$
\begin{aligned}
&\mathcal{A}^*(x, y, z) \\
&\left[\begin{array}{l}
(x_*, y_*, z_*) \leftarrow \mathcal{R}^*(x, y, z) \\
\textbf{return } \mathcal{B}(x_*, y_*, z_*)
\end{array}\right. ,
\end{aligned}
$$

which works slightly differently from $\mathcal{A}$ when the input is non-DH tuple. More precisely, (x_*, y_*, z_*) is distributed uniformly over non-DH tuples instead of uniform distribution over $\mathbb{G} \times \mathbb{G} \times \mathbb{G}$. As the statistical distance between these distributions is $\frac{1}{q}$, we get $\mathsf{Adv}(\mathcal{A}) \geq \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(\mathcal{B}) - \frac{1}{q}$.

Amplification by majority voting. For clarity, we give the simplest construction where $\mathcal{B}$ is called out trice to amplify the distinguishing advantage. As before, let $\mathcal{A}$ denote the reduction algorithm for random self-reducibility. Then the new majority voting algorithm is the following:

$$
\begin{aligned}
&\mathcal{C}(x, y, z) \\
&\left[\begin{array}{l}
b_1 \leftarrow \mathcal{A}(x, y, z) \\
b_2 \leftarrow \mathcal{A}(x, y, z) \\
b_3 \leftarrow \mathcal{A}(x, y, z) \\
\textbf{return } [b_1 + b_2 + b_3 > 1]
\end{array}\right. .
\end{aligned}
$$

Analysis. The same advantage $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(\mathcal{B})$ can be achieved with different success probabilities

$$
\begin{aligned}
\varepsilon_0 &= \Pr\left[\mathcal{Q}_1^{\mathcal{B}} = 1\right] \\
\varepsilon_1 &= \Pr\left[\mathcal{Q}_1^{\mathcal{B}} = 1\right]
\end{aligned}
$$

as long as $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{B}) = |\varepsilon_1 - \varepsilon_0|$ and thus the analysis is not so straightforward as one might expect. If (x, y, z) is a DH tuple, then we know by previous analysis that

$$\Pr[b_i = 1] = \Pr[\mathcal{Q}_1^{\mathcal{B}} = 1] = \varepsilon_1$$

and thus

$$\Pr[\mathcal{C}(g^a, g^b, g^c) = 1 | c = ab] = \varepsilon_1^3 + 3\varepsilon_1^2(1 - \varepsilon_1) \enspace .$$

If (x, y, z) is not a DH tuple, then

$$\Pr[b_i = 1] = \Pr[\mathcal{Q}_0^{\mathcal{B}} = 1] = \varepsilon_0$$

and thus

$$\Pr\left[\mathcal{C}(g^a, g^b, g^c) = 1 | c \neq ab\right] = \varepsilon_0^3 + 3\varepsilon_0^2(1 - \varepsilon_0) \enspace .$$

By combining results

$$\begin{aligned}
\mathsf{Adv}(\mathcal{C}) &= \left|\varepsilon_0^3 + 3\varepsilon_1^2(1 - \varepsilon_1) - \varepsilon_0^3 + 3\varepsilon_0^2(1 - \varepsilon_0)\right| \\
&= \left|2(\varepsilon_1 - \varepsilon_0)(\varepsilon_1^2 + \varepsilon_1\varepsilon_0 + \varepsilon_0^2) - 3(\varepsilon_1 - \varepsilon_0)(\varepsilon_1 + \varepsilon_0)\right| \\
&= |\varepsilon_1 - \varepsilon_0| \cdot |3\varepsilon_0 + 3\varepsilon_1 - 2\varepsilon_0^2 - 2\varepsilon_0\varepsilon_1 - 2\varepsilon_0^2|
\end{aligned}$$

and thus

$$\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{C}) = \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{C}) \cdot |3\varepsilon_0 + 3\varepsilon_1 - 2\varepsilon_0^2 - 2\varepsilon_0\varepsilon_1 - 2\varepsilon_0^2| \enspace .$$

The last term can be lower bounded further if we assume that $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{A}) \in \left(\frac{1}{2}, \frac{3}{4}\right)$. Then

$$\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{C}) = \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{C})^2 \cdot (3 - 2 \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{A})) > \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{A}) \enspace .$$

and we indeed do get amplification of success probability. However, the gain is not so big and the derivation of the advantage is not so straightforward as it seems.

Further comments. The reduction construction $\mathcal{C}$ is not optimal for the analysis. Usually, one uses more complex indirect construction

$$\begin{array}{l}
\mathcal{C}(x, y, z) \\
\left[\begin{array}{l}
\overline{z} \xleftarrow[u]{} \mathbb{G} \\
i_1, i_1, i_3 \xleftarrow[u]{} \{0, 1\} \\
\mathsf{if}\ i_1 = 1\ \mathsf{then}\ b_1 \leftarrow \mathcal{A}(x, y, z)\ \mathsf{else}\ b_1 \leftarrow \mathcal{A}(x, y, \overline{z}) \\
\mathsf{if}\ i_2 = 1\ \mathsf{then}\ b_2 \leftarrow \mathcal{A}(x, y, z)\ \mathsf{else}\ b_2 \leftarrow \mathcal{A}(x, y, \overline{z}) \\
\mathsf{if}\ i_3 = 1\ \mathsf{then}\ b_3 \leftarrow \mathcal{A}(x, y, z)\ \mathsf{else}\ b_3 \leftarrow \mathcal{A}(x, y, \overline{z}) \\
\mathsf{if}\ [b_1 \stackrel{?}{=} i_1] + [b_2 \stackrel{?}{=} i_2] + [b_3 \stackrel{?}{=} i_3] > 1\ \mathsf{then}\ \mathbf{return}\ 1 \\
\mathsf{else}\ \mathbf{return}\ 0
\end{array}\right.
\end{array}$$

which allows to reduce the analysis on the analysis of biased coin throws and use Chebyshev or Hoeffding bounds to estimate the success of majority voting.