ETH Zürich, D-INFK
Spring 2018

Prof. Ueli Maurer
Christian Badertscher
Fabio Banfi

Cryptography Foundations
Solution Exercise 9

9.1 Random Self-Reducibility of the Computational Diffie-Hellman Problem

Let $\mathbb{G} = \langle g \rangle$ be a finite cyclic group of order $q := |\mathbb{G}|$, let $\mathcal{G}$ be the set of CDH game instances over $\mathbb{G}$ (i.e., systems $\mathbf{G}$ which output a pair (g^a, g^b), for any $a, b \in \mathbb{Z}_q$, and are won on input g^{ab}), and let G be the uniform distribution over $\mathcal{G}$. We devise a system $\mathbf{R}$ that transforms a fixed instance $\mathbf{g} \in \mathcal{G}$ to a uniformly random instance, i.e., $\mathbf{Rg} \equiv \mathbf{G}$ for all $\mathbf{g} \in \mathcal{G}$. The system $\mathbf{R}$ is defined as follows:

- Given an instance $(g_1, g_2) \in \mathbb{G}^2$ at the right interface, choose $c, d \in \mathbb{Z}_q$ uniformly at random and output the instance $(h_1, h_2) := (g_1 \cdot g^c, g_2 \cdot g^d)$ at the left interface.
- On input a candidate solution h_3 for the randomized instance at the left interface, compute the candidate solution $g_3 = h_3 \cdot g_1^{-d} \cdot g_2^{-c} \cdot g^{-ab}$ for the fixed instance and output it at the right interface.

Note that the instances generated by $\mathbf{R}$ are indeed uniformly random. It remains to show that $\mathbf{R}$ reconstructs a solution for the original instance if and only if it obtains a solution for the randomized instance. This is the case since if $g_1 = g^a$, $g_2 = g^b$, and $h_3 = h^{(a+b)(b+d)} = h^{ab+ad+bc+cd}$, we have $g_3 = h_3 \cdot g_1^{-d} \cdot g_2^{-c} \cdot g^{-ab} = g^{ab}$; if h_3 is not a correct solution, g_3 is also not correct, because multiplication by $g_1^{-d} \cdot g_2^{-c} \cdot g^{-ab}$ is injective.

9.2 Cloning the MAC-forgery Game

a) We have to show that $\mathbf{KG}_{\mathsf{mac}} \not\equiv \left(\mathbf{G}_{\mathsf{mac}}^{[q]}\right)^{\vee}$. To this end, consider the winner $\mathbf{W}$ that queries in the first instance some message $m \in \mathcal{M}$ and obtains the MAC $t = f(m, k)$. Then, $\mathbf{W}$ inputs the pair (m, t) as a forgery in the second instance.

Note that (m, t) counts as a forgery in the second instance of $\left(\mathbf{G}_{\mathsf{mac}}^{[q]}\right)^{\vee}$, where no message was queried before. Hence, $\overline{\left(\mathbf{G}_{\mathsf{mac}}^{[q]}\right)^{\vee}}(\mathbf{W}) = 1$. In contrast, the system $\mathbf{K}$ submits the forgery to the same instance of $\mathbf{G}_{\mathsf{mac}}$ that was used to obtain the tag $t = f(m, k)$. Therefore, it is not counted as a forgery, so $\overline{\mathbf{KG}_{\mathsf{mac}}}(\mathbf{W}) = 0$. We conclude that $\overline{\left(\mathbf{G}_{\mathsf{mac}}^{[q]}\right)^{\vee}}(\mathbf{W}) \neq \overline{\mathbf{KG}_{\mathsf{mac}}}(\mathbf{W})$ and in particular $\mathbf{KG}_{\mathsf{mac}} \not\equiv \left(\mathbf{G}_{\mathsf{mac}}^{[q]}\right)^{\vee}$, so the system $\mathbf{K}$ does not clone the game $\mathbf{G}_{\mathsf{mac}}$.

b) Let $\mathbf{K}$ be the "straightforward" cloning system that simply forwards the inputs and outputs, as in subtask a). Note that a strategy similar to $\mathbf{W}$ from subtask a) does not apply in the game $\left(\mathbf{G}_{\mathsf{fix}}^{[q]}\right)^{\vee}$ since querying $\hat{m}$ is forbidden in every instance of $\left(\mathbf{G}_{\mathsf{fix}}^{[q]}\right)^{\vee}$.

We now show that $\mathbf{KG}_{\mathsf{fix}} \equiv \left(\mathbf{G}_{\mathsf{fix}}^{[q]}\right)^{\vee}$. In the beginning, both systems output the same message $\hat{m}$ in each of the q clones. Then, on input a message $m \in \mathcal{M}$ in any of the clones, both systems compute and return $t = f(m, k)$ using the same (and in the two

cases equivalently distributed) key in all clones (or, if $m = \hat{m}$, both systems return $\perp$). Finally, on input a tag $\hat{t} \in \mathcal{T}$, both systems set the MBO 1 if and only if $\hat{t} = f(\hat{m}, k)$. Hence, the systems $\mathbf{KG}_{\mathsf{fix}}$ and $\left(\mathbf{G}_{\mathsf{fix}}^{[q]}\right)^{\vee}$ have the same behavior.

The important property of $\mathbf{G}_{\mathsf{fix}}$ that we have exploited is that, given the instance $(k, \hat{m})$, the system $\mathbf{G}_{\mathsf{fix}}$ is stateless.

9.3 Performance Amplification Revisited

Let $\pi \coloneqq \rho^{\mathbf{K}}\sigma^2$. We then have for all winners $\mathbf{W}$ with $\mathbb{E}\big[X_{\mathbf{W}}^2\big] < \mathbb{E}\big[X_{\mathbf{W}}\big]$,

$$\begin{aligned}
\overline{\mathbf{G}}\pi(\mathbf{W}) &= \overline{\mathbf{G}}\rho^{\mathbf{K}}(\mathbf{W}^2) \\
&= \overline{\mathbf{KG}}(\mathbf{W}^2) \\
&= \overline{\mathbf{G}^{[2]^{\vee}}}(\mathbf{W}^2) \\
&= \Pr^{\mathbf{WG}}\Big[\omega\Big(\mathbf{W}^2, \mathbf{G}^{[2]^{\vee}}\Big) = 1\Big] \\
&= \sum_{\mathbf{g}\in\mathcal{G}} \Pr^{\mathbf{G}}[\mathbf{G} = \mathbf{g}] \cdot \Pr^{\mathbf{W}}\Big[\omega\Big(\mathbf{W}^2, \mathbf{G}^{[2]^{\vee}}\Big) = 1 \;\Big|\; \mathbf{G} = \mathbf{g}\Big] \\
&= \sum_{\mathbf{g}\in\mathcal{G}} \Pr^{\mathbf{G}}[\mathbf{G} = \mathbf{g}] \cdot \Big(1 - \big(1 - \Pr^{\mathbf{W}}[\omega(\mathbf{W}, \mathbf{g}) = 1]\big)^2\Big) \\
&= \sum_{\mathbf{g}\in\mathcal{G}} \Pr^{\mathbf{G}}[\mathbf{G} = \mathbf{g}] \cdot \Big(2 \cdot \Pr^{\mathbf{W}}[\omega(\mathbf{W}, \mathbf{g}) = 1] - \Pr^{\mathbf{W}}[\omega(\mathbf{W}, \mathbf{g}) = 1]^2\Big) \\
&= 2 \cdot \mathbb{E}\big[X_{\mathbf{W}}\big] - \mathbb{E}\big[X_{\mathbf{W}}^2\big] \\
&> \mathbb{E}\big[X_{\mathbf{W}}\big] \\
&= \overline{\mathbf{G}}(\mathbf{W}).
\end{aligned}$$

9.4 Properties of the Distinguishing Advantage

Let $S_1, S_2, S_3 \in \mathcal{O}$ probabilistic objects. The equality $\Delta^{\mathcal{D}}(S_1, S_1) = 0$ holds since $\Delta^{D}(S_1, S_1) = 0$ for all $D \in \mathcal{D}$. Since $\mathcal{D}$ is closed under the complement, we have that $\Delta^{\mathcal{D}}(S_1, S_2)$ is non-negative. Therefore, $\Delta^{\mathcal{D}}(S_1, S_2) = \sup_{D\in\mathcal{D}}|\Delta^{D}(S_1, S_2)|$ and moreover, by the definition of the distinguishing advantage we have that $|\Delta^{D}(S_1, S_2)| = |\Delta^{D}(S_2, S_1)|$ for all D. Hence,

$$\Delta^{\mathcal{D}}(S_1, S_2) = \sup_{D\in\mathcal{D}} |\Delta^{D}(S_1, S_2)| = \sup_{D\in\mathcal{D}} |\Delta^{D}(S_2, S_1)| = \Delta^{\mathcal{D}}(S_2, S_1).$$

In addition, we have that:

$$\begin{aligned}
\Delta^{\mathcal{D}}(S_1, S_3) &= \sup_{D\in\mathcal{D}} \Delta^{D}(S_1, S_3) \\
&= \sup_{D\in\mathcal{D}} \big(\Pr^{DS_1}[\kappa(D, S_1) = 1] - \Pr^{DS_3}[\kappa(D, S_3) = 1]\big) \\
&= \sup_{D\in\mathcal{D}} \Big(\Pr^{DS_1}[\kappa(D, S_1) = 1] - \Pr^{DS_2}[\kappa(D, S_2) = 1] \\
&\qquad + \Pr^{DS_2}[\kappa(D, S_2) = 1] - \Pr^{DS_3}[\kappa(D, S_3) = 1]\Big) \\
&\leq \sup_{D\in\mathcal{D}} \big(\Pr^{DS_1}[\kappa(D, S_1) = 1] - \Pr^{DS_2}[\kappa(D, S_2) = 1]\big) \\
&\quad + \sup_{D\in\mathcal{D}} \big(\Pr^{DS_2}[\kappa(D, S_2) = 1] - \Pr^{DS_3}[\kappa(D, S_3) = 1]\big) \\
&= \Delta^{\mathcal{D}}(S_1, S_2) + \Delta^{\mathcal{D}}(S_2, S_3).
\end{aligned}$$

9.5 Abstract Models of Computation

a) The model is described by a black box $\mathbf{B}$ with internal variables $V_i \in \{0,1\}^\ell$, such that V_1 is initialized to the secret value $x \in \{0,1\}^\ell$ chosen uniformly at random. The allowed operations are the nullary operations π_y for $y \in \{0,1\}^\ell$ that simply set the next (j-th) variable to $V_j := y$ and the binary group operation $\pi_\oplus(m,n)$ that sets $V_j = V_m \oplus V_n$. The unary inversion $\pi_{\mathsf{inv}}(i)$ can be omitted, since it can be obtained by adding $0 \in \{0,1\}^\ell$ to the variable V_i. The only allowed query is the binary query $\sigma_=(m,n)$ that returns 1 if and only if $V_m = V_n$.

b) The algorithm proceeds similar to the baby-step-giant-step (BSGS) algorithm.

First, it prepares a "runway" of size $2^{\ell/2}$, for instance by calling π_y for all $y \in \{0,1\}^\ell$ with the first ℓ bits being 0, resulting in $2^{\ell/2}$ calls to $\mathbf{B}$.

In the second phase, the algorithm iteratively generates the $2^{\ell/2}$ elements $z \in \{0,1\}^\ell$ where the second half of the bits is 0, and tests whether the elements computed by $x \oplus z$ are on the "runway", by comparing it to the $2^{\frac{\ell}{2}}$ values y generated in step 1. This process is guaranteed to succeed after at most $2^{\ell/2}$ passes (for one of the values z, the first half of the bits will coincide with the first half of x, which means that the first half of $x \oplus z$ is 0, which means that $x \oplus z$ is on the "runway"). The value x can then be easily reconstructed from z and $x \oplus z$.

Overall, the algorithm performs $\mathcal{O}(2^{\ell/2})$ operations and $\mathcal{O}(2^\ell)$ queries in the worst case (analogous to BSGS in the reading assignment).

c) One could imagine a new type of query according to the arbitrary total ordering $\preceq$: The binary query $\sigma_\preceq(m,n)$ returns 1 if and only if $V_m \preceq V_n$.

Similar to the BSGS algorithm, this query can be used to improve the high number of equality checks in the above algorithm. Indeed, the algorithm could implement a much faster check whether a value computed in step 2 equals one of the $2^{\frac{\ell}{2}}$ equidistant runway points y generated in step 1: we build a sorted list by applying merge-sort to the $2^{\frac{\ell}{2}}$ values y generated in step 1. This takes no more $O(2^{\frac{\ell}{2}} \cdot k)$ steps. Searching for a concrete element in this list boils down to perform binary search in this list and hence step 2 gives an overall number of queries of at most $2^{\frac{\ell}{2}} \cdot \frac{\ell}{2}$.

We therefore get the overall number of steps of $\mathcal{O}(\ell \cdot 2^{\frac{\ell}{2}})$.

d) The Theorem from the reading assignment we invoke is:

Theorem 4.8.3 *Let $\star$ be a group operation on S, let $\Pi = \mathbf{Const} \cup \{x \to x \star a \,|\, a \in S\}$ consists of all constant functions and group actions, and let $\Sigma = \{=\}$. Then the success probability of every k-step algorithm for extraction is at most $\frac{1}{4}(k+1)^2/|S|$.*

Due to the similarity of the above approach to the baby-step-giant-step algorithm, the considerations of Section 4.8.3 of the reading assignment apply: to get constant success probability we need the number of operations to be in the order $\mathcal{O}(\sqrt{|S|}) = \mathcal{O}(2^{\frac{\ell}{2}})$. So, with respect to the number of operations that our algorithm performs[1] the proposed approach seems optimal in this restricted model of computation.

The last thing to justify is why we invoked Theorem 4.8.3. The reason is the following: any value that can be computed (from the initial state) with the given operations can be expressed as functions of the form $x^a \star c$ where a and c are known constants, and the term x^a is (as usual) shorthand for $x \star \cdots \star x$ (a times). Note that the group we consider is actually the product group $\mathbb{Z}_2^\ell$ with group operation $\star$ being the element-wise addition modulo 2.

[1] Recall that the lower bounds in the reading assignment do only count the number of operations but not the number of queries the algorithm performs.

Hence, only two cases can occur: if $a \equiv_2 0$ then the function $x^a \star c$ is an element of **Const** because XOR'ing x an even number of times with itself yields 0^k. If $a \equiv_2 1$ then the function just corresponds to the group action $x \star c$.

Hence, each computed function by an algorithm A using the above set of operations can be computed by an algorithm B using just operations in Π and in addition, B does not need more steps than A to do that (and is thereby equally successful). Hence, the upper bound on the success probability of every k-step algorithm for extraction (with respect to class Π) applies to our problem using this reduction argument and invoking Theorem 4.8.3.