ETH Zürich, D-INFK
Spring 2019

Prof. Ueli Maurer
Fabio Banfi
Daniel Jost
Jiamin Zhu

Cryptography Foundations
Solution Exercise 5

5.1 Abstract Models of Computation

a) The model is described by a black box **B** with internal variables $v_i \in \{0,1\}^k$, such that V_1 is initialized to the secret value $x \in \{0,1\}^k$ chosen uniformly at random. The allowed operations are the nullary operations π_y for $y \in \{0,1\}^k$ 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 $0 \in \{0,1\}^k$ to the variable V_i.

There is one type query allowed, the binary equality query; given a query $(=,i,j)$ with $i,j \in \mathbb{N}$, the box **B** returns 1 if and only if $V_i = V_j$.

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

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

In the second phase, the algorithm iteratively generates the $2^{k/2}$ elements $z \in \{0,1\}^k$ 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 $2^{k/2}$ values y generated in step 1. This process is guaranteed to succeed after at most $2^{k/2}$ 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 $O(2^{k/2})$ operations and $O(2^k)$ 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 $\preceq$. Given a query $(\preceq, i, j)$ with indices $i,j \in \mathbb{N}$, **B** returns 1 if and only if $V_i \preceq V_j$.

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 $2^{k/2}$ equidistant runway points y generated in step 1: we build a sorted list by applying merge-sort to the $2^{k/2}$ values y generated in step 1. This takes no more than $O(2^{k/2} \cdot k)$ 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 $2^{k/2} \cdot \frac{k}{2}$.

We therefore get the overall number of steps of $O(k \cdot 2^{k/2})$.

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

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

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 t to be in the order $O(\sqrt{|S|}) = O(2^{\frac{k}{2}})$. 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 $\mathbb{Z}_2^k$ with group operation $\star$ being the element-wise addition modulo 2.

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 t-step algorithm for extraction (with respect to class Π) applies to our problem using this reduction argument and invoking Theorem 4.8.3.

5.2 The Proof of Theorem 4.8.4

a) Let $Q(x_1, \ldots, x_t)$ be a non-zero multivariate polynomial over $\mathbb{Z}_{q^e}$. Let $f \leq e$ be the largest exponent such that q^f divides every coefficient of Q. Observe that since Q is a non-zero polynomial (modulo q^e) we have $0 \leq f < e$. Now let Q' denote the multivariate polynomial obtained by dividing each coefficient of Q by q^f, i.e, such that $q^f \cdot Q' = Q$. Hence, we have

$$q^e \mid (q^f \cdot Q') \iff q^{e-f} \mid Q', \tag{1}$$
$$q^{e-f} \mid Q' \implies q \mid Q'. \tag{2}$$

Using this, we derive that

$$\begin{aligned}
&\left|\{(x_1, \ldots, x_t) \in \mathbb{Z}_{q^e}^t \mid Q(x_1, \ldots, x_t) \equiv_{q^e} 0\}\right| \\
&\overset{(1)}{=} \left|\{(x_1, \ldots, x_t) \in \mathbb{Z}_{q^e}^t \mid Q'(x_1, \ldots, x_t) \equiv_{q^{e-f}} 0\}\right| \\
&\overset{(2)}{\leq} \left|\{(x_1, \ldots, x_t) \in \mathbb{Z}_{q^e}^t \mid Q'(x_1, \ldots, x_t) \equiv_q 0\}\right| \\
&= q^{(e-1)t} \cdot \left|\{(x_1, \ldots, x_t) \in \mathbb{Z}_q^t \mid Q'(x_1, \ldots, x_t) \equiv_q 0\}\right| \\
&\leq q^{(e-1)t} \cdot \frac{d}{q} \cdot q^t \\
&= d \cdot q^{et-1},
\end{aligned}$$

where the second last step follows by Lemma 4.8.1, observing that by definition of Q' (and f) there must be at least one coefficient of Q' that is not divisible by q. Thus, Q' is non-zero with modulo q. Overall, we get the desired result as

$$\frac{\left|\{(x_1, \ldots, x_t) \in \mathbb{Z}_{q^e}^t \mid Q(x_1, \ldots, x_t) \equiv_{q^e} 0\}\right|}{\left|\mathbb{Z}_{q^e}^t\right|} \leq \frac{d \cdot q^{et-1}}{q^{et}} = \frac{d}{q}.$$

b) Let $\ell = \frac{n}{q^e}$. Since $\gcd(q^e, k) = 1$, we have by the Chinese Remainder Theorem that $\mathbb{Z}_n \cong \mathbb{Z}_{q^e} \times \mathbb{Z}_\ell$. This implies that we can emulate the black-box $\mathbf{B}$ for $\mathbb{Z}_n$ from a black-box $\mathbf{B}_1$ for $\mathbb{Z}_{q^e}$ and a black-box $\mathbf{B}_2$ for $\mathbb{Z}_\ell$, as depicted in Figure 1.

[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.

Recall that we say that the generic DL solver $\mathcal{A}$ is successful if and only if he manages to input the value x of the first register into any other register as a constant. Hence, in our emulation of $\mathbf{B}$, if the solver $\mathcal{A}$ is successful, then he in particular successfully extracted from the black-box $\mathbf{B}_1$. Hence, we can transform any generic solver $\mathcal{A}$ into a generic solver $\mathcal{A}'$ (that emulates $\mathcal{A}$ and the black-box $\mathbf{B}_2$ internally), that performs the same number of steps and has a success probability at least as good as the one by $\mathcal{A}$.

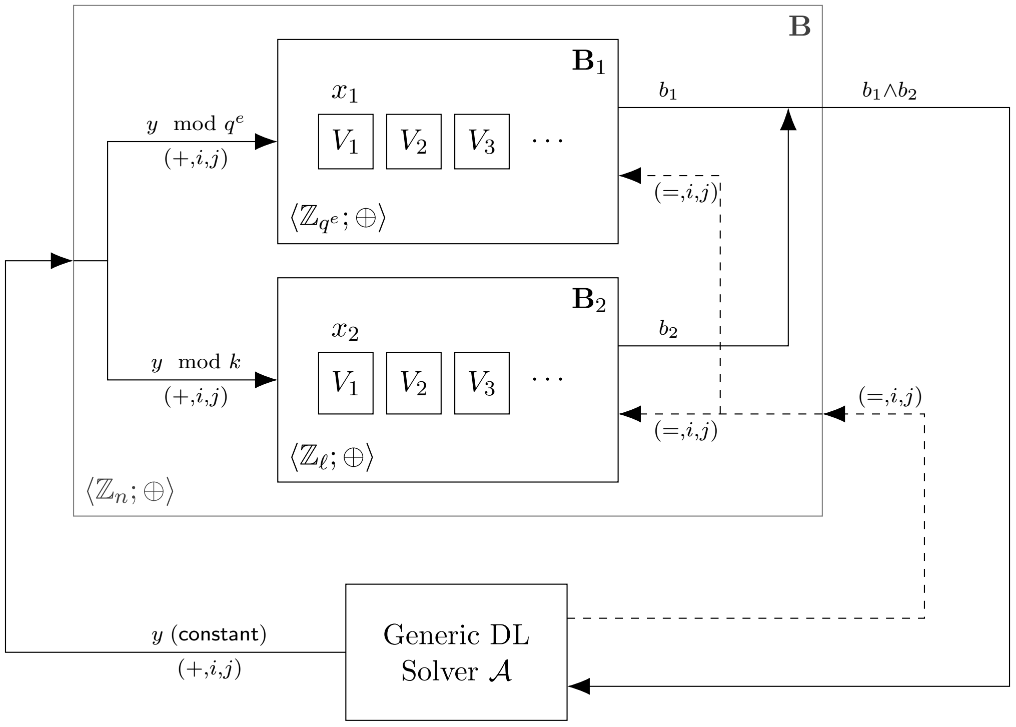

Figure 1: Illustration of how to emulate the black-box $\mathbf{B}$ for $\mathbb{Z}_n$ from a black-box $\mathbf{B}_1$ for $\mathbb{Z}_{q^e}$ and a black-box $\mathbf{B}_2$ for $\mathbb{Z}_\ell$.

c) Let q denote the largest prime factor of n and e denotes its multiplicity. Recall from **b)** that it is sufficient to upper bound the probability of any k-step DL solver for $\mathbb{Z}_{q^e}$. To bound this success probability, we follow the same strategy as in the reading assignment. First, consider the closure of the operation set $\overline{\Pi} := \{x \mapsto ax + b \mid a, b \in \mathbb{Z}_{q^e}\}$. Then, recall that solving the extraction problem implies provoking a non-trivial collision. Hence, we can upper bound the success probability by the probability of provoking a non-trivial collision, and thus we can restrict ourselves to non-adaptive solvers.

Any non-adaptive strategy corresponds to choosing k different linear functions $(a_i x + b_i) \in \overline{\Pi}$. Using the lemma proved in **a)**, for $i \neq j$ we can bound the probability of

$$a_i x + b_i \equiv_{q^e} a_j x + b_j \quad \Longleftrightarrow \quad (a_i - a_j)x + (b_i - b_j) \equiv_{q^e} 0,$$

with $\frac{1}{q}$, since we have non-zero polynomial of degree $d = 1$. Since there are $k+1$ many different values $a_i x + b_i$ (including $1x + 0$ in the first register), there are $\binom{k+1}{2}$ many such pairs. Using the union bound, we can derive the overall bound on the success probability:

$$\binom{k+1}{2}\frac{1}{q} \leq \frac{1}{2}(k+1)^2/q.$$

d) For the DDH problem our argument from subtask **b)** does not work. For the extraction problem, inputing a constant c such that $c = x \pmod{n}$ implies that $c \equiv_{q^e} x \wedge c \equiv_\ell x$

for *every* prime factor q. In contrast, in order to distinguish the two settings in the DDH problem, it is sufficient to detect any difference, so a difference either with respect to q^e or with respect to ℓ. All we can say is that there must exists *some* prime factor with respect to which he detects some difference. Hence, we have to do a worst-case analysis and take the smallest prime factor.

5.3 Generic Reduction of the DL Problem to the CDH Problem

a) As suggested in the reading assignment (Section 4.8.7), we need to consider the extraction problem for the ring $\mathbb{Z}_p$ with additive operation $\oplus$ (addition modulo p) and multiplicative operation $\odot$ (multiplication modulo p). The model is described by a black box $\mathbf{B}$ with internal variables $V_i \in \mathbb{Z}_p$, such that V_1 is initialized to the secret value $x \in \mathbb{Z}_p$ chosen uniformly at random (this corresponds to the discrete logarithm of an element of the group $\mathbb{G}$). The allowed operations are the nullary operations π_y for $y \in \mathbb{Z}_p$ that simply set the next (j-th) variable to $V_j := y$, the binary operation $\pi_\oplus(m,n)$ that sets $V_j := V_m \oplus V_n$, and the binary operation $\pi_\odot(m,n)$ that sets $V_j := V_m \odot V_n$. The only allowed query is the binary query $\sigma_=(m,n)$ that returns 1 if and only if $V_m = V_n$.

b) Let h be a generator of $\mathbb{Z}_p^*$ that is assumed to be known. Then the reduction is specified by the three converters $\mathbf{C}_\Pi$, $\mathbf{C}_\Sigma$, and $\mathbf{C}_{\mathsf{out}}$.

- On input the operation π_y (with $y \in \mathbb{Z}_{p-1}$) at the outside of $\mathbf{C}_\Pi$, the operation π_{h^y} (with $h^y \in \mathbb{Z}_p^*$) is output at the inside of $\mathbf{C}_\Pi$. On input the operation $\pi_\oplus(m,n)$ at the outside of $\mathbf{C}_\Pi$, the operation $\pi_\odot(m,n)$ is output at the inside of $\mathbf{C}_\Pi$.
- On input the query $\sigma_=(m,n)$ at the outside of $\mathbf{C}_\Sigma$, the query $\sigma_=(m,n)$ is output at the inside of $\mathbf{C}_\Sigma$.
- On input the value $w \in \mathbb{Z}_{p-1}$ at the inside of $\mathbf{C}_{\mathsf{out}}$, the value $h^w \in \mathbb{Z}_p^*$ is output at the outside of $\mathbf{C}_{\mathsf{out}}$.

Brief justification of this reduction (recall the ideas from the lecture): the reduction simply changes the internal representation of the values and inputs of $\mathcal{A}$ via an isomorphism $\phi_h : \mathbb{Z}_{p-1} \to \mathbb{Z}_p^*$, i.e., $\phi_h(y) = h^y$. Thanks to this homomorphism, we have $\phi(x+y) = h^x \odot h^y$, (which is to say that we represent the addition as a multiplication), and thanks to the bijective property, the equality queries are answered correctly. Hence, the algorithm $\mathcal{A}$ which solves the extraction problem for (any element of) the additive group $\mathbb{Z}_{p-1}$ can be used to compute the correct result for the extraction problem for the multiplicative group $\mathbb{Z}_p^*$.

c) After we have shown how to translate (efficiently) any algorithm $\mathcal{A}$ that solves the extraction problem for $\mathbb{Z}_{p-1}$ to an algorithm that solves the extraction problem for $\mathbb{Z}_p^*$, we invoke the results from the reading assignment (or exercise 3.3): the Pohlig-Hellman algorithm[2] described in Section 4.6.3 of the reading assignment (coupled with BSGS as indicated there) requires $O(\sqrt{q'}\log n)$ operations where q' is the largest prime factor of $n := |\mathbb{Z}_{p-1}| = p-1$ and by B-smoothness we have $q' \leq B$. If B is treated as a constant, we have $O(\log n)$, i.e., the runtime grows linear in the input size (which are elements of a group of size n).

This illustrates the basic idea behind the reduction mentioned in Section 4.8.7 of the reading assignment. The full result is found in the references given in Section 4.8.7 (which is not part of this course) and considers elliptic curves instead of the simpler objects $\mathbb{Z}_p^*$ to find a good reduction.

[2]We assume that the factorization of $p-1$ is known.