Swiss Post Public Intrusion Test

Generating Random Group Elements
(Best Practice)

Rolf Haenni

Bern University of Applied Sciences

1 Description of Issue

Sometimes, it is necessary to select a group element uniformly at random from a prime-order subgroup $G_q \subset \mathbb{Z}_p^*$ of the integers modulo p, where $q = |G_q|$ is the order of the subgroup and $h = \frac{p-1}{q}$ the corresponding *co-factor*. The proposed algorithm in [1, Section 9.1.26] and the corresponding method `getRandomElement` in the class `GroupTools` first select a random element r from $\mathbb{Z}_q \setminus \{0\} = \{1, \ldots, q-1\}$ and then return $H = g^r \bmod p$, where $g \in G_q$ is a generator of the subgroup.[1] While this method generates random group elements with equal probability (except for $H = 1$, see footnote), it has a potentially unwanted side-effect. During the computation of H, the discrete logarithm $r = \log_g H \bmod p$ is known to the machine that executes the algorithm and therefore may leak to an adversary. Depending on the purpose of H, this may be completely unproblematic, but in some cases, the security of a cryptographic primitive that uses H may be at risk. For example, if two supposedly independent generators $H_1, H_2 \in G_q$ are selected in this way, then knowledge of $r_1 = \log_g H_1 \bmod p$ and $r_2 = \log_g H_2 \bmod p$ reveals $\log_{H_1} H_2 \bmod p = \frac{r_2}{r_1} \bmod q$, i.e., the require independence of the generators is broken. If H_1, H_2 are then used for example in a Pedersen commitment scheme, then the computational binding property is no longer guaranteed, i.e., the scheme is entirely broken. This example shows that generating random group elements as suggested can lead to unwanted mistakes for no real benefit.

2 Solution

For $h = 2$ there are two best practices for generating random elements from G_q. The first selects a random element $r \in \mathbb{Z}_p^*$ and then checks $r \in G_q$ by testing $r^q \bmod p = 1$

[1]Excluding $r = 0$ implies that $H = 1$ is selected with probability 0 and all other elements are selected with probability $\frac{1}{q-1}$. This is slightly different from selecting elements uniformly at random with equal probability $\frac{1}{q}$. We recommend therefore selecting r from $\mathbb{Z}_q = \{0, \ldots, q-1\}$.

(or more efficiently using the Legendre symbol $(\frac{r}{p}) = 1$). If yes, r is returned, otherwise the process is starts from the beginning. The second method also selects a random element $r \in \mathbb{Z}_p^*$, but then returns $r^2 \bmod p$. In both cases, all elements of G_q are returned with equal probability $\frac{1}{q}$. Note that the second solution is the most efficient one. It requires only a single modular multiplication (squaring), which is more than three orders of magnitude more efficient than a modular exponentiation for the given bit lengths. Note that the second method can be easily generalized to the general case of $h \geq 2$ by returning $r^h \bmod p$.

References

[1] Scytl sVote – Protocol Specifications. Technical report, Scytl Secure Electronic Voting, Barcelona, Spain, 2018.