Advanced cryptography: Cloud & More, winter 2013/14

MICHAEL NÜSKEN

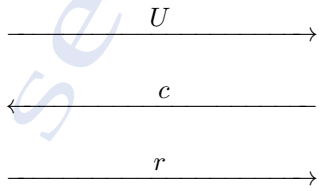
10. Exercise sheet
Hand in solutions until Wednesday, 29 January 2014, 23:59

Exercise 10.1 (Zero-Knowledge). (18 points)

Consider the following interactive protocol:

Protocol EqDlog. Interactive proof of knowledge of a discrete logarithm.
Publicly known: El Gamal parameters (G, q, P) and a challenge set $S \subset \mathbb{Z}_q$.
Public input: Group elements $P, T \in G$.
Private input to the prover: The discrete logarithm $t \in \mathbb{Z}_q$ of T wrt. P, ie. $T = P^t$.

1. The prover chooses a temporary private key $u \xleftarrow{\$} S \subset \mathbb{Z}_q$ and computes $U \leftarrow P^u$ in G. She sends U to the verifier.
2. The verifier chooses a challenge $c \xleftarrow{\$} S$ and sends it to the prover.
3. The prover computes the response $r \leftarrow u + ct$ and sends it to the verifier.
4. The verifier checks that $P^r = UT^c$.



The ElGamal parameter are a group G (for example given as an implementation), a generator P and its order ℓ. The security parameter n is given by $n = \lceil \log_2 \ell \rceil$. The group is assumed to be efficient, ie. the algorithms for the group operation(s) are polynomial time wrt. n. Further, we assume that ℓ is prime. The challenge set S is understood to be a function of ℓ, say $S = \{x \bmod q \in \mathbb{Z}_q \mid x \in \mathbb{N}, x < n^{17}\}$ or $S = \{0, 1\}$. It has always at least 2 and at most poly(n) elements. Finally, we assume that discrete logarithms in G are hard to find.

(i) Show that the prover Paula and the verifier Victor each run in polynomial time regardless of what the other does. *Note*: Clearly, Paula needs her private input to be able to run in polynomial time. [2]

(ii) Show that Protocol EqDlog is complete. [2]

(iii) Show that Protocol EqDlog is sound with error at most $1/\#S$. [3]

(iv) Show that Protocol EqDlog is honest-verifier zero-knowledge. *Hint*: the simulator Sammy has roughly the same runtime as Victor. [3]

(v) Show that Protocol EqDlog is zero-knowledge with a simulator Sam having expected runtime essentially $\#S$ times the runtime of Victor, which is poly(n). *Hint*: You may want to use Sammy as a building block here. [4]

(vi) Show that Protocol EqDlog is a proof-of-knowledge. *Hint*: The needed knowledge extractor will ask two challenges c, c' for one U. [4]