- it may request implicit certificates from the CA
- finally, it produces an implicit certificate and the corresponding private key in time t and with probability p.

We define an impersonating adversary A_i as a probabilistic, polynomial-time turing machine which, on input (PK_{CA}, SK_{CA}) does the following:

- it may act as a CA and issue implicit certificates to requesting entities
- it can produce an implicit certificate and the corresponding private key in time t and with probability p.

An adversary A_f (respectively, A_i) is successful if t is polynomial and p is non-negligible.

8.4 Unforgeability

Let (x, Y) be the (SK, PK) key pair of the CA (represented through shareholders in case of the implicit certificate scheme). Let $D_{NormSchnorr}$ denote Schnorr's signature scheme and $A_{NormSchnorr}$ be a successful adversary against it as defined earlier in Section 5.1. We define a successful adversary $A_{DistCert}$ against the implicit certificate scheme $D_{DistCert}$ as a successful forging adversary as defined in Section 8.3.

One can show that a successful adversary $A_{NormSchnorr}$ is equivalent to a successful adversary $A_{DistCert}$, in the sense that each of them can construct the other one. This implies that the distributed implicit certificate scheme is as secure as Schnorr's signature scheme.

The same proof technique as was used for the distributed Schnorr signature scheme can be applied in a straightforward way. That is, one can show how to simulate the view of the given adversary without knowing the private key of the shareholders. Since the adversary cannot distinguish a simulated view from an actual view, he will perform his attack and output a forgery. This forgery can then be used to construct the other adversary.

8.5 Non-impersonating

By proving the unforgeability of our scheme, we implicitly proved that the user does not learn the shareholders' private key shares. We also have to show that the shareholders do not learn the user's private key and impersonate the user. But it follows directly from the scheme that if the shareholders could compute the user's private key, then they could compute discrete logarithms. $\Box$

8.6 Further Issues

Consider the scenario where a digital signature on a certain message and an implicit certificate authenticating the according verification key are sent to a user. Even though we proved that it is hard to forge an implicit certificate without knowing the CA's secret key such that one knows the corresponding private key, we did not prove that it is hard to forge a digital signature and an implicit certificate such that the public key implied by the certificate just validates the signature.