(Proof) We show how to construct $A_{NormSchnorr}$ given the adversary $A_{DistSchnorr}$. Informally, we will show how $A_{NormSchnorr}$ can simulate – with the help of a signing oracle (used in the chosen message attack assumption) – the role of the honest players in $D_{DistSchnorr}$ for a given public key Y. Because $A_{DistSchnorr}$ cannot distinguish this simulation, it will be successful and output a forgery which is a forgery in $D_{NormSchnorr}$, too.

For simplicity, assume $A_{DistSchnorr}$ corrupts players $1, ..., t-1$. Using the techniques described in Lemma 1, $A_{NormSchnorr}$ lets *SIM* execute the key generation protocol for the given public key Y. Next, $A_{NormSchnorr}$ runs $A_{DistSchnorr}$. Whenever $A_{DistSchnorr}$ requests a signature for m_i, $A_{NormSchnorr}$ asks a signer and provides $A_{DistSchnorr}$ with the signature (m_i, σ_i, V_i). $A_{NormSchnorr}$ also has to provide $A_{DistSchnorr}$ with the values he sees during the signature issuing protocol. These values include, in particular, the view resulting from generating a random shared secret and all the $\{\gamma_i\}$. Again, $A_{NormSchnorr}$ lets *SIM* interact with $A_{DistSchnorr}$ during the generation of a random shared secret. As a side effect, *SIM* (and therefore also $A_{NormSchnorr}$) knows $\alpha_1, ..., \alpha_{t-1}, \beta_1, ..., \beta_{t-1}$ and can compute $\gamma_1, ..., \gamma_{t-1}$. Finally, $A_{NormSchnorr}$ computes γ_t as follows. Recall from Section 4.2 that we have

$$\sigma_i = \sum_{j=1}^{t} \gamma_j \omega_j, \text{ where } \omega_j = \prod_{\substack{h=1 \\ h \neq j}}^{t} \frac{h}{h-j}.$$

Hence, γ_t is computed as

$$\gamma_t = \frac{\sigma_i - \sum_{j=1}^{t} \gamma_j \omega_j}{\omega_t}.$$

Now $A_{NormSchnorr}$ feeds $\{\gamma_1, ..., \gamma_t\}$ to $A_{DistSchnorr}$. Since $A_{DistSchnorr}$ now has his whole view, he can perform his adaptive chosen message attack. $A_{NormSchnorr}$ outputs $(\tilde{m}, \tilde{\sigma}, \tilde{V})$ if $A_{DistSchnorr}$ outputs $(\tilde{m}, \tilde{\sigma}, \tilde{V})$. □

6 The Implicit Certificate Scheme

To motivate the (t, n) threshold scheme for implicit certificates, we give a short overview of the non-distributed version of this scheme ([3]). In [3], security proofs for this scheme in the random oracle model are given.

Assume a CA with the key pair (x, Y) issues an implicit certificate to a user. The operation of the scheme is as follows.

1. The user generates a random integer $c \in Z_q$ and computes $V = cG$. Further, he sends V to the CA.
2. The CA authenticates the user. Together, the CA and the user determine an identifier string I_u (containing the user's identity and other information such as, for example, a serial number for the certificate).
3. The CA chooses a random integer $e \in Z_q$, and computes $C = V + eG$ and $\sigma = e + h(I_u, C)x$. Further, the CA sends (I_u, C, σ) to the user.
4. The user computes his private key $SK_u = c + s \bmod q$, and verifies the certificate by checking that following equation holds: $SK_u = C + h(I_u, C)Y$.