For convenience, we introduce the following notation for this protocol:

$$(s_1, ..., s_n) \quad \overset{(t,n)}{\longleftrightarrow} \quad (r|Y, a_iG, H_0),\ i \in \{1, ..., t-1\}.$$

This notation means that s_j is player P_j's share of the secret r for each $j \in H_0$. The values a_iG are the public commitments of the sharing polynomial $f(\cdot)$ (they can be computed using public information), and (r, Y) forms a *key pair* (i.e., r is a private key and Y is the corresponding public key). The set H_0 denotes the set of players that have not been detected to be cheating. In the further protocols, we do not need the values $C_i, Y_j, C_{ji}, s'_j, r'$ for $j \in \{1, ..., n\}, i \in \{0, ..., t-1\}$ and therefore we omit these values in the short notation.

3 Schnorr's Signature Scheme

In [10], Schnorr introduced the following signature scheme. Let (x, Y) be a user's key pair, let m be a message and let G be a generator of an elliptic curve group having prime order q. Then a user generates a Schnorr signature on the message m as follows.

1. Select $e \in Z_q$ at random
2. Compute $V = eG$
3. Compute $\sigma = e + h(m, V)x \bmod q$
4. Define the signature on m to be (V, σ)

A verifier accepts a signature (V, σ) on a message m if and only if $\sigma \in Z_q$ and

$$\sigma G = V + h(m, V)Y$$

Schnorr signatures were shown to be existentially unforgeable under adaptively chosen message attacks in the random oracle model, using the forking lemma in [10], provided that the discrete logarithm problem is hard in the group generated by G.

4 A (t, n) Threshold Signature Scheme

In this section, we propose a robust and efficient (t, n) threshold digital signature scheme for Schnorr signatures. We use the primitives presented in Section 2.

Our protocol consists of a key generation protocol and a signature issuing protocol. Let $P_1, ..., P_n$ be a set of signers and let G be a generator of an elliptic curve group of order q.

4.1 Key Generation Protocol

All n signers have to co-operate to generate a public key, and a secret key share for each P_j. They generate a random shared secret according to the protocol presented in Section 2.4. Let the output of the protocol