schemes (see Section 4.4). We give an example using the Com function just introduced. Assume there are n voters $V_1, \ldots, V_n$. For simplicity, we assume that only "yes-no" votes are possible. A trusted center T is needed to compute the outcome of the election. The center T is assumed to be honest. If T was dishonest, it could determine each voter's vote. Let E_T and D_T be ElGamal encryption and decryption functions for the trusted center T. To vote on a subject, each voter V_i chooses $m_i \in \{0,1\}$ as his vote, a random $r_i \in \{0, \ldots, q-1\}$ and computes $c_i := \mathrm{Com}(r_i, m_i)$. Then he broadcasts c_i to the public and sends $E_T(g^{r_i})$ to the trusted center T. T computes

$$D_T(\prod_{i=1}^{n} E_T(g^{r_i})) = \prod_{i=1}^{n} g^{r_i} = g^r,$$

where $r = \sum_{i=1}^{n} r_i$, and broadcasts g^r.

Now, everyone can compute the result s of the vote from the publicly known c_i, $i = 1, \ldots, n$, and g^r:

$$v^s = g^{-r} \prod_{i=1}^{n} c_i \,,$$

with $s := \sum_{i=1}^{n} m_i$. s can be derived from v^s by computing $v, v^2, \ldots$ and comparing with v^s in each step, because the number of voters is not too large. If the trusted center is honest, the factor g^{r_i} – which hides V_i's vote – is never computed. Although an unconditional hiding commitment is used, the hiding property is only computational because g^{r_i} is encrypted with a cryptosystem that provides at most computational security.

4.4 Electronic Elections

In an *electronic voting scheme* there are two distinct types of participants: the voters casting the votes and the voting authority (for short, the authority) that collects the votes and computes the final tally.

Usually the following properties are required: (1) *universal verifiability* ensures that the correctness of the election, especially the correct computation of the tally, can be checked by everyone; (2) *privacy* ensures that the secrecy of an individual vote is maintained; and (3) *robustness* ensures that the scheme works even in the presence of a coalition of parties with faulty behavior. Naturally, only authorized voters should be allowed to cast their votes.

There seems to be a conflict between the last requirement and privacy. The scheme we describe resolves this conflict by establishing a group of authorities and a secret sharing scheme. It guarantees privacy, even if some of the authorities collaborate.

4.4.1 Secret Sharing

The idea of secret sharing is to start with a secret s and divide it into n pieces called shares. These shares are distributed among n users in a secure way. A coalition of some of the users is able to reconstruct the secret. The secret could, for example, be the password to open a safe shared by five people, with any three of them being able to reconstruct the password and open the safe.

In a (t, n)-*threshold scheme* $(t \le n)$, a trusted center computes the shares s_j of a secret s and distributes them among n users. Each t of the n users are able to recover s from their shares. It is impossible to recover the secret from $t-1$ or fewer shares.

Shamir's Threshold Scheme. Shamir's threshold scheme is based on the following properties of polynomials over a finite field k. For simplicity, we take $k = \mathbb{Z}_p$, where p is a prime.

Proposition 4.13. *Let* $f(X) = \sum_{i=0}^{t-1} a_i X^i \in \mathbb{Z}_p[X]$ *be a polynomial of degree* $t-1$*, and let* $P := \{(x_i, f(x_i)) \mid x_i \in \mathbb{Z}_p,\ i = 1, \ldots, t, x_i \neq x_j, i \neq j\}$. *For* $Q \subseteq P$*, let* $\mathcal{P}_Q := \{g \in \mathbb{Z}_p[X] \mid deg(g) = t-1, g(x) = y \text{ for all } (x, y) \in Q\}$.

1. $\mathcal{P}_P = \{f(X)\}$*, i.e.,* f *is the only polynomial of degree* $t-1$*, whose graph contains all* t *points in* P.
2. *If* $Q \subset P$ *is a proper subset and* $x \neq 0$ *for all* $(x, y) \in Q$*, then each* $a \in \mathbb{Z}_p$ *appears with the same frequency as the constant coefficient of a polynomial in* $\mathcal{P}_Q$.

Proof. To find all polynomials $g(X) = \sum_{i=0}^{t-1} b_i X^i \in \mathbb{Z}_p[X]$ of degree $t-1$ through m given points $(x_i, y_i), 1 \le i \le m$, you have to solve the following linear equations:

$$\begin{pmatrix} 1 & x_1 & \ldots & x_1^{t-1} \\ \vdots & & & \\ 1 & x_m & \ldots & x_m^{t-1} \end{pmatrix} \begin{pmatrix} b_0 \\ \vdots \\ b_{t-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

If $m = t$, then the above matrix (called A) is a Vandermonde matrix and its determinant

$$\det A = \prod_{1 \le i < j \le t} (x_j - x_i) \neq 0,$$

if $x_i \neq x_j$ for $i \neq j$. Hence,the system of linear equations has exactly one solution and part 1 of Proposition 4.13 follows.

Now let $Q \subset P$ be a proper subset. Without loss of generality, Q consists of the points $(x_1, y_1), \ldots, (x_m, y_m)$, $1 \le m \le t-1$. We consider the following system of linear equations:

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & x_1 & \dots & x_1^{t-1} \\ \vdots & & & \\ 1 & x_m & \dots & x_m^{t-1} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{t-1} \end{pmatrix} = \begin{pmatrix} a \\ y_1 \\ \vdots \\ y_m \end{pmatrix} \tag{4.1}$$

The matrix of the system consists of rows of a Vandermonde matrix (note all $x_i \neq 0$ by assumption). Thus, the rows are linearly independent and the system (4.1) has solutions for all $a \in \mathbb{Z}_p$. The matrix has rank $m+1$ independent of a. Hence, the number of solutions is independent from a, and we see that each $a \in \mathbb{Z}_p$ appears as the constant coefficient of a polynomial in $\mathcal{P}_Q$ with the same frequency. □

Corollary 4.14. *Let $f(X) = \sum_{i=0}^{t-1} a_i X^i \in \mathbb{Z}_p[X]$ be a polynomial of degree $t-1$, and let $P = \{(x_i, f(x_i)) \mid i = 1, \dots, t, x_i \neq x_j, i \neq j\}$. Then*

$$f(X) = \sum_{i=1}^{t} f(x_i) \prod_{1 \leq j \leq t, j \neq i} (X - x_j)(x_i - x_j)^{-1}. \tag{4.2}$$

This formula is called the Lagrange interpolation formula.

Proof. The right-hand side of (4.2) is a polynomial g of degree $t-1$. If we substitute X by x_i in g, we get $g(x_i) = f(x_i)$. Since the polynomial $f(X)$ is uniquely defined by P, the equality holds. □

Now we describe Shamir's (t, n)-threshold scheme. A trusted center T distributes n shares of a secret $s \in \mathbb{Z}$ among n users $P_1, \dots, P_n$. To set up the scheme, the trusted center T proceeds as follows:

1. T chooses a prime $p > \max(s, n)$ and sets $a_0 := s$.
2. T selects $a_1, \dots, a_{t-1} \in \{0, \dots, p-1\}$ independently and at random, and gets the polynomial $f(X) = \sum_{i=0}^{t-1} a_i X^i$.
3. T computes $s_i := f(i)$, $i = 1, \dots, n$ (we use the values $i = 1, \dots, n$ for simplicity; any n pairwise distinct values $x_i \in \{1, \dots, p-1\}$ could also be used) and transfers (i, s_i) to the user P_i in a secure way.

Any group of t or more users can compute the secret. Let $J \subset \{1, \dots, n\}$, $|J| = t$. From Corollary 4.14 we get

$$s = a_0 = f(0) = \sum_{i \in J} f(i) \prod_{j \in J, j \neq i} j(j-i)^{-1} = \sum_{i \in J} s_i \prod_{j \in J, j \neq i} j(j-i)^{-1}.$$

If only $t-1$ or fewer shares are available, then each $a \in \mathbb{Z}_p$ is equally likely to be the secret (by Proposition 4.13). Thus, knowing only $t-1$ or fewer shares provides no advantage over knowing none of them.

Remarks:

1. Suppose that each $a \in \mathbb{Z}_p$ is equally likely as the secret for someone knowing only $t-1$ or fewer shares, as in Shamir's scheme. Then the (t, n)-threshold scheme is called *perfect*: the scheme provides perfect secrecy in the information-theoretic sense (see Section 9.1). The security does not rely on the assumed hardness of a computational problem.
2. Shamir's threshold scheme is easily extendable for new users. New shares may be computed and distributed without affecting existing shares.
3. It is possible to implement varying levels of control. One user can hold one or more shares.

4.4.2 A Multi-Authority Election Scheme

For simplicity, we only consider election schemes for yes-no votes. The voters want to get a majority decision on some subject. In the scheme that we describe, each voter selects his choice (yes or no), encrypts it with a homomorphic encryption algorithm and signs the cryptogram. The signature shows that the vote is from an authorized voter. The votes are collected in a single place, the bulletin board. After all voters have posted their votes, an authority can compute the tally without decrypting the single votes. This feature depends on the fact that the encryption algorithm used is a homomorphism. It guarantees the secrecy of the votes. But if the authority was dishonest, she could decrypt the single votes. To reduce this risk, the authority consists of several parties and the decryption key is shared among these parties, by use of a Shamir (t, n)-threshold scheme. Then at least t of the n parties must be dishonest to reveal a vote. First, we assume in our discussion that a trusted center T sets up the scheme. However, the trusted center is not really needed. In Section 4.4.6 we show that it is possible to set up the scheme by a communication protocol which is executed by the parties constituting the authority.

The election scheme that we describe was introduced in [CraGenSch97]. The time and communication complexity of the scheme is remarkably low. A voter simply posts a single encrypted message, together with a compact proof that it contains a valid vote.

The Communication Model. The members of the voting scheme communicate through a *bulletin board*. The bulletin board is best viewed as publicly accessible memory. Each member has a designated section of the memory to post messages. No member can erase any information from the bulletin board. The complete bulletin board can be read by all members (including passive observers). We assume that a public-key infrastructure for digital signatures is used to guarantee the origin of posted messages.

Setting up the Scheme. To set up the scheme, we assume for now that there is a trusted center T. The trusted center T chooses primes p and q, such that q is a large divisor of $p-1$, and an element $g \in \mathbb{Z}_p^*$ of order q, as in the key generation procedure of the DSA (see Section 3.5.3). g generates the subgroup G_q of order q of $\mathbb{Z}_p^*$.[5]

Further, we assume that T chooses a secret key $s \in \{0, \ldots, q-1\}$ at random and publishes the public key $h := g^s$ to be used for ElGamal encryption with respect to the base g. A message $m \in G_q$ is encrypted as $(c_1, c_2) = (g^\alpha, h^\alpha m)$, where α is a randomly chosen element in $\{0, \ldots, q-1\}$. Using the secret key s the plaintext m can be recovered as $m = c_2 c_1^{-s}$ (see Section 3.5.1). The encryption is homomorphic: if (c_1, c_2) and (c_1', c_2') are encryptions of m and m', then

$$(c_1, c_2) \cdot (c_1', c_2') = (c_1 c_1', c_2 c_2') = (g^\alpha g^{\alpha'}, h^\alpha m h^{\alpha'} m') = (g^{\alpha+\alpha'}, h^{\alpha+\alpha'} m m')$$

is an encryption of mm'.

Let $A_1, \ldots, A_n$ be the authorities and $V_1, \ldots, V_m$ be the voters in the election scheme. The trusted center T chooses a Shamir (t, n)-threshold scheme. The secret encryption key s is shared among the n authorities. A_j keeps her share (j, s_j) secret. The values $h_j := g^{s_j}$ are published on the bulletin board by the trusted center.

Decryption. Everyone can decrypt a cryptogram $c := (c_1, c_2) := (g^\alpha, h^\alpha m)$ with some help of the authorities, but without reconstructing the secret key s. Namely, the following steps are executed:

1. Each authority A_j posts $w_j := {c_1}^{s_j}$ to the bulletin board. Here, we assume that the authority A_j is honest and follows the protocol. In Section 4.4.3 we will see how to check that she really does (by a proof of knowledge).
2. Let J be the index set of a subset of t honest authorities. Then, everyone can recover $m = c_2 c_1^{-s}$ as soon as all A_j, $j \in J$, have finished step 1:

$$ {c_1}^s = {c_1}^{\sum_{j \in J} s_j \lambda_{j,J}} = \prod_{j \in J} ({c_1}^{s_j})^{\lambda_{j,J}} = \prod_{j \in J} {w_j}^{\lambda_{j,J}}, $$

where

$$\lambda_{j,J} = \prod_{l \in J \setminus \{j\}} l(l-j)^{-1} .$$

Vote Casting. Each voter V_i selects his vote $v_i \in \{-1, 1\}$, encodes v_i as g^{v_i} and encrypts g^{v_i} by the ElGamal encryption:

$$c_i := (c_{i,1}, c_{i,2}) := (g^{\alpha_i}, h^{\alpha_i} g^{v_i}).$$

He then signs it to guarantee the origin of the message and posts it to the bulletin board. Here we assume that V_i follows the protocol and correctly forms c_i. He has to perform a proof of knowledge which shows that he really does (see Section 4.4.3); otherwise his vote is invalid.

[5] There is a unique subgroup of order q of $\mathbb{Z}_p^*$. It is cyclic and each element $x \in \mathbb{Z}_p^*$ of order q is a generator (see Lemma A.40 and "Computing modulo a prime" on page 303).

Tally Computing. Assume that m votes were cast:

1. Everyone can compute
$$c = (c_1, c_2) = \left(\prod_{i=1}^{m} c_{i,1}, \prod_{i=1}^{m} c_{i,2} \right).$$
Note that $c = (c_1, c_2)$ is the encryption of g^d, where d is the difference between the number of yes votes and no votes, since the encryption is homomorphic.
2. The decryption protocol from above is executed to get g^d. After sufficiently many authorities A_j have posted $w_j = c_1^{s_j}$ to the bulletin board, everyone can compute g^d.
3. Now d can be found by computing the sequence $g^{-m}, g^{-m+1}, \ldots$, and comparing with g^d in each step.

Remarks:

1. Everyone can check whether a voter or an authority was honest (see Section 4.4.3), and discard invalid votes. If he finds a subset of t honest authorities, he can compute the tally. This implies universal verifiability.
2. No coalition of $t-1$ or fewer authorities can recover the secret key. This guarantees the robustness of the scheme.
3. Privacy depends on the security of the underlying ElGamal encryption scheme and, hence, on the assumed difficulty of the Diffie-Hellman problem. The scheme provides only computational privacy. A similar scheme is introduced in [CraFraSchYun96] which even provides perfect privacy (in the information-theoretic sense). This is achieved by using a commitment scheme with information-theoretic secure hiding to encrypt the votes.
4. The following remarks concern the communication complexity of the scheme:
   a. Each voter only has to send one message together with a compact proof that the message contains a valid vote (see below). His activities are independent of the number n of authorities.
   b. Each authority has to read m messages from the bulletin board, verify m interactive proofs of knowledge and post one message to the bulletin board.
   c. To compute the tally, you have to read t messages from the bulletin board and to verify t interactive proofs of knowledge.
5. It is possible to prepare an election beforehand. The voter V_i chooses $v_i \in \{-1, 1\}$ at random. The voting protocol is executed with the random v_i values. Later, the voter decides the alternative to choose. He selects $\tilde{v}_i \in \{-1, 1\}$, such that $\tilde{v}_i v_i$ is his vote, and posts $\tilde{v}_i$ to the bulletin board. The tally is computed with $\tilde{c}_i = (c_{i,1}^{\tilde{v}_i}, c_{i,2}^{\tilde{v}_i}), i = 1, \ldots, m$.

4.4.3 Proofs of Knowledge

Authority's Proof. In the decryption protocol above, each authority A_j has to prove that she really posts $w_j = c_1^{s_j}$, where s_j is her share of the secret key s. Recall that $h_j = g^{s_j}$ is published on the bulletin board. The authority has to prove that w_j and h_j have the same logarithm with respect to the bases c_1 and g and that she knows this logarithm. We simplify the notation and describe an interactive proof of knowledge of the common logarithm x of $y_1 = g_1^x$ and $y_2 = g_2^x$, where x is a random element from $\{0, \ldots, q-1\}$. As usual, we call the prover Peggy and the verifier Vic.

Of course, in our voting scheme it is desirable for practical reasons that the authority proves, in a non-interactive way, to be honest. However, it is easy to convert the interactive proof into a non-interactive one (see Section 4.4.4). Thus, we first give the interactive version of the proof.

Protocol 4.15.
ProofLogEq(g_1, y_1, g_2, y_2):

1. Peggy chooses $r \in \{0, \ldots, q-1\}$ at random and sets $a := (a_1, a_2) = (g_1^r, g_2^r)$. Peggy sends a to Vic.
2. Vic chooses $c \in \{0, \ldots, q-1\}$ uniformly at random and sends c to Peggy.
3. Peggy computes $b := r - cx$ and sends b to Vic.
4. Vic accepts if and only if $a_1 = g_1^b y_1^c$ and $a_2 = g_2^b y_2^c$.

The protocol is a three-move protocol. It is very similar to the protocol used in the simplified Fiat-Shamir identification scheme (see Section 4.2.2):

1. The first message is a commitment by Peggy. She commits that two numbers have the same logarithm with respect to the different bases g_1 and g_2.
2. The second message c is a challenge by Vic.
3. The third message is Peggy's response. If $c = 0$, Peggy has to open the commitment (reveal r). If $c \neq 0$, Peggy has to show her secret in encrypted form (reveal $r - cx$).

Completeness. If Peggy knows a common logarithm for y_1 and y_2, and both Peggy and Vic follow the protocol, then $a_1 = g_1^b y_1^c$ and $a_2 = g_2^b y_2^c$, and Vic will accept.

Soundness. A cheating prover Eve can convince Vic with a probability of $1/q$ in the following way:

1. Eve chooses $r, \tilde{c} \in \{0, \ldots, q-1\}$ at random, sets $a := (g_1^r y_1^{\tilde{c}}, g_2^r y_2^{\tilde{c}})$ and sends a to Vic.
2. Vic chooses $c \in \{0, \ldots, q-1\}$ at random and sends c to Eve.
3. Eve sends r to Vic.

Vic accepts if and only if $c = \tilde{c}$. The event $c = \tilde{c}$ occurs with a probability of $1/q$. Thus, Eve succeeds in cheating with a probability of $1/q$.

If Eve can convince Vic with a probability greater than $1/q$ (the probability is taken over the challenges c), she has to answer at least two challenges correctly, for a given commitment a.

Suppose Eve knows an $a = (a_1, a_2)$ for which she can answer two distinct challenges c and $\tilde{c}$. This means that Eve can compute b and $\tilde{b}$, such that

$$\begin{aligned} a_1 &= g_1^b y_1^c, \quad a_2 = g_2^b y_2^c, \\ a_1 &= g_1^{\tilde{b}} y_1^{\tilde{c}}, \quad a_2 = g_2^{\tilde{b}} y_2^{\tilde{c}}. \end{aligned}$$

Then she can compute

$$g_1^{\tilde{b}-b} = y_1^{c-\tilde{c}} \text{ and } g_2^{\tilde{b}-b} = y_2^{c-\tilde{c}}$$

and gets

$$(\tilde{b} - b)(c - \tilde{c})^{-1} = \log_{g_1}(y_1) \text{ and } (\tilde{b} - b)(c - \tilde{c})^{-1} = \log_{g_2}(y_2).$$

Thus, she can compute the secret x. This contradicts the assumption that it is infeasible to compute x from g^x (for randomly chosen p, g and x). We see that the probability of success of a cheating prover is bounded by $1/q$.

Honest Verifier Zero-Knowledge. The above protocol is not known to be zero-knowledge. However, it is zero-knowledge if the verifier is an honest one. An interactive proof system (P, V) is called *honest-verifier zero-knowledge* if Definition 4.6 holds for the honest verifier V, but not necessarily for an arbitrary Verifier V^*; i.e., there is a simulator S that produces correctly distributed accepting transcripts for executions of the protocol with (P, V).

To simulate an interaction with the honest verifier is quite simple. The key point is that the honest verifier V chooses the challenge $c \in \{0, \ldots, q-1\}$ independently from (a_1, a_2), uniformly and at random, and this can also be done by S.

Algorithm 4.16.

```
int $S$(int $g_1, g_2, y_1, y_2$)
1  select $\tilde{b} \in \{0, \ldots, q-1\}$ uniformly at random
2  select $\tilde{c} \in \{0, \ldots, q-1\}$ uniformly at random (this is $V$'s task)
3  $\tilde{a_1} \leftarrow g_1^{\tilde{b}} y_1^{\tilde{c}}, \tilde{a_2} \leftarrow g_2^{\tilde{b}} y_2^{\tilde{c}}$
4  return $(\tilde{a}_1, \tilde{a}_2, \tilde{c}, \tilde{b})$
```

The transcript $(\tilde{a}_1, \tilde{a}_2, \tilde{c}, \tilde{b})$ returned by S is an accepting transcript and not distinguishable from a transcript (a_1, a_2, c, b) produced by (P, V):

1. a_1, a_2 and $\tilde{a}_1, \tilde{a}_2$ are randomly chosen elements in $\mathbb{Z}_q$.
2. c and $\tilde{c}$ are randomly chosen elements in $\{0, \ldots, q-1\}$.
3. b and $\tilde{b}$ are randomly chosen elements in $\{0, \ldots, q-1\}$.

Voter's Proof. In the vote-casting protocol, each voter has to prove that he really encrypted a vote $g^v \in \{g, g^{-1}\}$; i.e., he has to prove that $c = (c_1, c_2) = (g^\alpha, h^\alpha m)$ and $m \in \{g, g^{-1}\}$. For this purpose, he performs a proof of knowledge. He proves that he knows α for either $c_1 = g^\alpha$ and $c_2 g^{-1} = h^\alpha$, or for $c_1 = g^\alpha$ and $c_2 g = h^\alpha$. Each of the two alternatives could be proven as in the authority's proof. Here, however, the prover's task is more difficult. The proof must not reveal which of the two alternatives is proven. An interactive three-move proof that convinces the verifier without revealing anything about the prover's choice is the subject of Exercise 9.

4.4.4 Non-Interactive Proofs of Knowledge

It is easy to convert an interactive three-move proof into a non-interactive one using the standard method of Fiat-Shamir, which we demonstrated in Section 4.2.5. Let $h : \{0,1\}^* \longrightarrow \mathbb{Z}_q$ be a collision-resistant hash function. We get a non-interactive proof

$$(c, b) = \mathrm{ProofLogEq}_h(g_1, y_1, g_2, y_2),$$

for $y_1 = g_1^x$ and $y_2 = g_2^x$ in the following way. The prover Peggy chooses $r \in \{0, \ldots, q-1\}$ at random and sets $a := (a_1, a_2) = (g_1^r, g_2^r)$. Then she computes the challenge $c := h(g_1 \| y_1 \| g_2 \| y_2 \| a_1 \| a_2)$ and sets $b := r - cx$. The verification condition is

$$c = h(g_1 \| y_1 \| g_2 \| y_2 \| g_1^b y_1^c \| g_2^b y_2^c).$$

The verifier needs not know a to compute the verification condition. If we trust the collision resistance of h, we can conclude that $u = v$ from $h(u) = h(v)$.

If we convert our proofs of knowledge into non-interactive proofs, honest-verifier zero-knowledge is sufficient, because here, the verifier is always honest. In our election protocol, each authority and each voter completes his message with a non-interactive proof which convinces everyone that he followed the protocol.

4.4.5 Extension to Multi-Way Elections

We describe how to extend the scheme if a choice between several, say $l > 2$, options should be possible.

To encode the votes $v_1, \ldots, v_l$, we independently choose l generators g_j, $j = 1, \ldots, l$, of G_q, and encode v_j by g_j. Voter V_i encrypts his vote v_j as

$$c_i = (c_{i,1}, c_{i,2}) = (g^\alpha, h^\alpha g_j).$$

Each voter shows – by an interactive proof of knowledge – that he knows α with

$$c_{i,1} = g^{\alpha} \text{ and } g_j^{-1} c_{i,2} = h^{\alpha},$$

for exactly one j. We refer the interested reader to [CraGenSch97] for some hints about the proof technique used.

The problem of computing the final tally turns out to be more complicated. The result of the tally computation is

$$c_2 c_1^{-s} = g_1^{d_1} g_2^{d_2} \dots g_l^{d_l},$$

where d_j is the number of votes for v_j. The exponents d_j are uniquely determined by $c_2 c_1^{-s}$ in the following sense: computing a different solution $(\tilde{d}_1, \dots, \tilde{d}_l)$ would contradict the discrete logarithm assumption, because the generators were chosen independently (see Proposition 4.21 in Section 4.5.3). Again as above, $d = (d_1, \dots, d_l)$ can be found for small values of l and d_i by searching.

4.4.6 Eliminating the Trusted Center

The trusted center sets up an ElGamal cryptosystem, generates a secret key, publishes the corresponding public key and shares the secret key among n users using a (t, n)-threshold scheme. To eliminate the trusted center, all these activities must be performed by the group of users (in our case, the authorities). For the communication between the participants, we assume below that a bulletin board, mutually secret communication channels and a commitment scheme exist.

Setting Up an ElGamal Cryptosystem. We need large primes p, q, such that q is a divisor of $p - 1$, and a generator g of the subgroup G_q of order q of $\mathbb{Z}_p^*$. They are generated jointly by the group of users (in our case the group of authorities). This can be achieved if each party runs the same generation algorithm (see Section 3.5.1). The random input to the generation algorithm, must be generated jointly. To do this, the users $P_1, \dots, P_n$ execute the following protocol:

1. Each user P_i chooses a string r_i of random bits of sufficient length, computes a commitment $c_i = C(r_i)$ and posts the result to the bulletin board.
2. After all users have posted their commitments, each user opens his commitment.
3. They take $r = \oplus_{i=1}^{n} r_i$ as the random input.

Publish the Public Key. To generate and distribute the public key the users $P_1, \dots, P_n$ execute the following protocol:

1. Each user P_i chooses $x_i \in \{0, \dots, q - 1\}$ at random, computes $h_i := g^{x_i}$ and a commitment $c_i := C(h_i)$ for h_i. Then he posts c_i to the bulletin board.

2. After all users have posted their commitments, each user opens his commitment.
3. Everyone can compute the public key $h := \prod_{i=1}^{n} h_i$.

Share the Secret Key. The corresponding secret key

$$x := \sum_{i=1}^{n} x_i$$

must be shared. The basic idea is that each user constructs a Shamir (t, n)-threshold scheme to share his part x_i of the secret key. These schemes are combined to get a scheme for sharing x.

Let $f_i(X) \in \mathbb{Z}_p[X]$ be the polynomial of degree $t - 1$ used for sharing x_i, and let

$$f(X) = \sum_{i=1}^{n} f_i(X).$$

Then $f(0) = x$ and f can be used for a (t, n)-threshold scheme, provided $\deg(f) = t - 1$. The shares $f(j)$ can be computed from the shares $f_i(j)$:

$$f(j) = \sum_{i=1}^{n} f_i(j).$$

The group of users executes the following protocol to set up the threshold scheme:

1. The group chooses a prime $p > \max(x, n)$.
2. Each user P_i randomly chooses $f_{i,j} \in \{0, \dots, p - 1\}$, $j = 1, \dots, t - 1$, and sets $f_{i,0} = x_i$ and $f_i(X) = \sum_{j=0}^{t-1} f_{i,j} X^j$. He posts $F_{i,j} := g^{f_{i,j}}$, $j = 1, \dots, t - 1$, to the bulletin board. Note that $F_{i,0} = h_i$ is P_i's piece of the public key and hence is already known.
3. After all users have posted their encrypted coefficients, each user tests whether $\sum_{i=1}^{n} f_i(X)$ has degree $t - 1$, by checking
$$\prod_{i=1}^{n} F_{i,t-1} \neq [1].$$
In the rare case that the test fails, they return to step 2. If $f(X)$ passes the test, the degree of $f(X)$ is $t - 1$.
4. Each user P_i distributes the shares $s_{i,l} = f_i(l)$, $l = 1, \dots, n$, of his piece x_i of the secret key to the other users over secure communication channels.
5. Each user P_i verifies for $l = 1, \dots, n$ that the share $s_{l,i}$ received from P_l is consistent with the previously posted committed coefficients of P_l's polynomial:
$$g^{s_{l,i}} = \prod_{j=0}^{t-1} (F_{l,j})^{i^j}.$$
If the test fails, he stops the protocol by broadcasting the message

"failure,$(l, i), s_{l,i}$".

6. Finally, P_i computes his share s_i of x:

$$s_i = \sum_{l=1}^{n} s_{l,i},$$

signs the public key h and posts his signature to the bulletin board.

After all members have signed h, the group will use h as their public key. If all participants followed the protocol, the corresponding secret key is shared among the group. The protocol given here is described in [Pedersen91]. It only works if all parties are honest. [GenJarKraRab99] introduces an improved protocol which works if a majority of the participants is honest.

4.5 Digital Cash

The growth of electronic commerce in the Internet requires *digital cash*. Today, credit cards are used to pay on the Internet. Transactions are online; i.e., all participants – the customer, the shop and the bank – are involved at the same time (for simplicity, we assume only one bank). This requires that the bank is available even during peak traffic time, which makes the scheme very costly. Exposing the credit card number to the vendor provides him with the ability to impersonate the customer in future purchases. The bank can easily observe who pays which amount to whom and when, so the customer cannot pay anonymously, as she can with ordinary money.

A payment with ordinary money requires three different steps. First, the customer fetches some money from the bank, and his account is debited. Then he can pay anonymously in a shop. Later, the vendor can bring the money to the bank, and his account is credited.

Ordinary money has an acceptable level of security and functions well for its intended task. Its security is based on a complicated and secret manufacturing process. However, it is not as secure in the same mathematical sense as some of the proposed digital cash schemes.

Digital cash schemes are modeled on ordinary money. They involve three interacting parties: the bank, the customer and the shop. The customer and the shop have accounts with the bank. A digital cash system transfers money in a secure way from the customer's account to the shop's account. In the following, the money is called an *electronic coin*, or *coin* for short.

As with ordinary money, paying with digital cash requires three steps:

1. The customer fetches the coin from the bank: customer and bank execute the withdrawal protocol.
2. The customer pays the vendor: customer and vendor execute the payment protocol.