1 Parameters of the Encryption Scheme

- There are n authorities, $A_1 \dots A_n$.
- Let k be the minimum number of authorities required to jointly decrypt a cyphertext.
- Let p and q be large primes, where $p = 2q + 1$ (q is commonly called a Sophie Germain prime, p a safe prime). A pair of such numbers can be found by generating a random prime q and checking if $2q+1$ is also prime.

How do we show that's feasable? Prime number theorem?

- Let g be a generator of G_q, where G_q is the unique subgroup of $\mathbb{Z}_p^*$ of order q. The *Decisional Diffie–Hellman assumption* is believed to hold for G_q, as G_q is the subgroup of quadratic residues in $\mathbb{Z}_q^*$. [**?**]

Write down proof? Usually just stated as a fact in literature

- The generator g can be computed as follows [**?**, Section 4.6]:
  1. Repeatedly choose an $\alpha \in \mathbb{Z}_p^*$ at random, until it satisfies $\alpha^q \neq 1$ and $\alpha^2 \neq 1$, that is, the order of α is neither q, 2 nor 1. Then α is a generator of $\mathbb{Z}_p^*$.
     Proof: By Lagrange's Theorem, $\mathbb{Z}_p^*$ has exactly two proper non-trivial subgroups of order p and 2, respectively. As α is neither of order p, 2 nor 1, it can only be a generator of $\mathbb{Z}_p^*$.
  2. Compute $g = \alpha^k$, where $k = (p-1)/q$. Then g is a generator of G_q.
     Proof: Let $ord(\cdot)$ be the order a group element. As k divides $ord(\alpha)$, it follows from a standard result of group theory [**?**, Proposition 4.5] that $ord(\alpha^k) = ord(\alpha)/k = q$.

2 Key Distribution

- Let $x := \sum_{i=1}^{n} x_i$ be the private key. Note that no single authority should be able to know x.
- Every authority A_i chooses a random $x_i \in \mathbb{Z}_q$, and publishes $h_i := g^{x_i}$.
- Let $h := g^x$ is the public key, which can be computed as $h = \prod_{i=1}^{n} h_i$.
- Every authority A_i generates the random polynomial

$$f_i(z) = \sum_{l=0}^{k-1} f_{i,l}^{l}, \tag{1}$$

with $f_i(z) \in \mathbb{Z}_q[z]$, where $f_{i,0} = 0$ and $f_{i,l} \in \mathbb{Z}_q$ is chosen randomly for $l \neq 0$. It follows by definition that $f_i(0) = x_i$.

- Every authority A_i publishes $(F_{i,l})_{l=1,\dots,k-1}$, where

$$F_{i,l} = g^{f_{i,l}} \tag{2}$$

is the commitment of authority A_j to the value of $f_{i,l}$.

- Now every authority A_i secretly sends

$$s_{i,j} = f_i(j) \tag{3}$$

to each authority A_j.

- A_i verifies the share received from A_j is consistent with the previously published values by verifying that

$$g^{s_{i,j}} = \prod_{l=0}^{k-1} F_{jl}^{(i^l)}. \tag{4}$$

This equation follows directly from raising g to both sides of equation (3).

Do we need to prove the consistency? Doesn't it just follow from the fact that it is the same computation, only in the exponent of g?

I think it should be illustrated why this works / what happens with the polynomials

- A_i computes his share of x as $s_i = \sum_{j=1}^{n} s_{ji}$.
- Each authority A_i publishes

$$\sigma_i := g^{s_i} \tag{5}$$

as a commitment to the received share.

3 Cooperative Decryption

- The full private key can be restored by a set at least k cooperating authorities $\Lambda \subseteq \{A_1, \ldots, A_n\}$, $k \leq |\Lambda|$, for example by using Lagrange interpolation:

$$x = \sum_{A_j \in \Lambda} s_j \lambda_{j,\Lambda} \tag{6}$$

where the Lagrange coefficients are

$$\lambda_{j,\Lambda} := \prod_{\substack{A_l \in \Lambda \\ l \neq j}} \frac{l}{l-k}. \tag{7}$$

Note that this formula is only used for the derivation of the cooperative encryption process, and authorities never actually should cooperate to restore the public key x.

- To decrypt an ElGamal encryption $(c_1, c_2) = (g^y, h^y m)$ of the message $m \in G_q$, each authority A_j broadcasts $w_j = c_1^{s_j}$.
- To prove that an authority has computed w_j correctly, it has to prove in zero-knowledge that

$$s_j = \log_g \sigma_j = \log_{c_1} w_j,$$

in words that w_j has actually been computed with the authority's share.

- By raising c_1 to both sides of equation (6) and then dividing c_2 by both sides, we get

$$m = c_2 / \prod_{A_j \in \Lambda} w_j^{\lambda_{j,\Lambda}}.$$

4 Zero-knowledge-proof for discrete logarithms

- The Prover wants to prove

$$s_j = \log_g \sigma_j = \log_{c_1} w_j$$

without revealing the value of s_j.

- The Prover sends (g^β, c_1^β), with $\beta \in_R Z_q$
- The Verifier sends $c \in_R Z_q$
- The Prover sends $r = \beta + s_i c$
- The Verifier checks the two equalities

$$g^r = g^\beta \sigma^c$$

$$c_1^r = c_1^\beta w_i^c$$

This proof utilizes the fact that it is hard to compute g^{ab} from g and a without having b.

5 Casting a vote

- A vote has the form $(g^y, h^y G^b)$, where G is a generator of G_q (one could just use $G = q$), $b \in \{-1, 1\}$ denotes the value of the vote, and $y \in_R Z_q$.

6 Verifying a vote

The details on how this protocol can be constructed from the discrete log protocol can be found in [CDS94].

Voter: $v = 1$	Voter: $v = -1$		Verifier
$\alpha, w, r_1, d_1 \in_R Z_q$	$\alpha, w, r_2, d_2 \in_R Z_q$		
$x \leftarrow g^\alpha$	$x \leftarrow g^\alpha$		
$y \leftarrow h^\alpha G$	$y \leftarrow h^\alpha / G$		
$a_1 \leftarrow g^{r_1} x^{d_1}$	$y \leftarrow g^w$		
$b_1 \leftarrow h^{r_1} (yG)^{d_1}$	$b_1 \leftarrow g^w$		
$a_2 \leftarrow g^w$	$y \leftarrow g^{r_2} x^{d_2}$		
$b_2 \leftarrow h^w$	$b_2 \leftarrow h^{r_2} (y/G)^{d_2}$	$\xrightarrow{x,y,a_1,b_1,a_2,b_2}$	
$d_2 \leftarrow c - d_1$	$d_1 \leftarrow c - d_2$	$\xleftarrow{c}$	$c \in_R Z_q$
$r_2 \leftarrow w - \alpha d_2$	$r_1 \leftarrow w - \alpha d_1$	$\xrightarrow{d_1,d_2,r_1,r_2}$	$c \stackrel{?}{=} d_1 + d_2$
			$a_1 \stackrel{?}{=} g^{r1} x^{d_1}$
			$b_1 \stackrel{?}{=} h^{r1} (yG)^{d_1}$
			$a_2 \stackrel{?}{=} g^{r2} x^{d_2}$
			$b_2 \stackrel{?}{=} h^{r2} (y/G)^{d_2}$

7 Counting votes

- Let (x_i, y_i) be the vote casted by Voter V_i
- $(X, Y) = (\prod_{i=1}^{l} x_i, \prod_{i=1}^{l} y_i)$ is computed by all authorities.
- (X, Y) is decrypted cooperatively, obtaining G^T, where T is the outcome of the election.
- Let l be the number of votes. As $T \in \{-t, ..., t\}$ holds, the number of votes can be found by brute-force.

8 Notes on Notation

[CGS97]	[Ped91]	this document	source code
s	x	x	BigInteger x
–	x_i	x_i	BigInteger[] xParts; xParts[i]

A ElGamal

To encrypt a cyphertext $m \in G_q$, the sender chooses a random $y \in_R Z_q$ and sends the pair $(c_1, c_2) = (g^y, mh^y)$. The decrypt the cyphertext, the receiver recovers the plaintext as $c_2/c_1^x = (mh^y)/g^{yx} = (mh^y)/h^y = m$.

References

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '94, pages 174–187, London, UK, UK, 1994. Springer-Verlag.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proceedings of the 16th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'97, pages 103–118, Berlin, Heidelberg, 1997. Springer-Verlag.

[Ped91] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'91, pages 522–526, Berlin, Heidelberg, 1991. Springer-Verlag.