Distributed Key Generation and Decryption

Polyas GmbH

November 16, 2025

The distributed key generation (DKG) method used in our system follows — up to minor differences in notation — the variant of of Pedersen's distributed key generation protocol [TP91] described in [BNPW24].

1 Setup and notation

The protocol works for an ElGamal group of order q, with a fixed generator g.

We consider a setup with n tellers and a threshold t. We will use indices k, l to range over the set $I = \{1, \ldots, n\}$ of tellers and indices i, j to range over the set $T = \{0, \ldots, t-1\}$ (of polynomial coefficients).

2 Distributed Key Generation

Polynomial generation. Each teller k generates a random polynomial of order $t-1$, where $a_{k,i} \in \mathbb{Z}_q$:

$$p_k(x) = a_{k,0} + a_{k,1}x + \cdots + a_{k,t-1}x^{t-1} \tag{DKG.1}$$

and publishes its **coefficient commitments**

$$A_{k,i} = g^{a_{k,i}} \qquad (i = 0, \ldots, t-1) \tag{DKG.2}$$

along with non-interactive zero-knowledge proof of knowledge of $a_{k,i}$.

The **public key** is determined by the published commitmets $A_{k,0}$ as:

$$Y_0 = \prod_{k \in I} A_{k,0} \tag{DKG.3}$$

with the corresponding **secret key** $y_0 = \sum_{k \in I} a_{k,0} = p(0)$, where $p(x) = \sum_{k \in I} p_k(x)$.

Sharing. Each teller k shares the following data with each other teller l:

$$y_{k,l} = p_k(l) \tag{DKG.4}$$

Teller l checks the zero-knowledge proofs for the commitments published in step (DKG.2) and verifies the data received from teller k, by checking the equation

$$g^{y_{k,l}} = \prod_{i \in T} (A_{k,i})^{l^i} \tag{DKG.5}$$

Note: *This check verifies that (DKG.4) is consistent with the commitments of teller k published in (DKG.2), by evaluating $p_k(l)$ "in the exponent".*

If these checks fail, the teller aborts; otherwise, the teller computes its **secret key share**:

$$y_l = \sum_{k \in I} y_{k,l} \mod q \tag{DKG.6}$$

and the corresponding value

$$Y_l = g^{y_l}. \tag{DKG.7}$$

Note: *We expect $y_l = p(l)$ and hence $Y_l = g^{p(l)}$.*

3 Verifiable Distributed Decryption

Assume that the decryption is carried out by a set $D \subseteq I$ of tellers of size $d \geq t$. To decrypt a ciphertext (α, β), one needs to compute α^{y_0} (recall that y_0 is the private key). This is jointly done by the tellers in D as follows. Each teller l computes and publishes it's decryption share

$$\bar{\alpha}_l = \alpha^{y_l} \tag{VDD.1}$$

along with a zero-knowledge proof of knowledge of y_l which simultaneously satisfies (VDD.1) and (DKG.7). For checking this zero-knowledge proof, one computes Y_l from the published coefficient commitments by

$$Y_l = \prod_{k \in I, i \in T} (A_{k,i})^{l^i} \tag{VDD.2}$$

The decryption is finalized by computing:

$$\alpha^{y_0} = \prod_{k \in D} (\bar{\alpha}_k)^{\ell_k^0} \qquad \text{with} \quad \ell_k^0 = \prod_{m \in D \setminus \{k\}} \frac{m}{m-k} \tag{VDD.3}$$

Note: *Note that $L(0) = \sum_{k \in D} y_k \ell_k^0$ is the value of the* Lagrange interpolating polynomial *(see [Lagr] for the set of nodes D and the corresponding values $\{y_k\}_{k \in D}$, evaluated in point 0. We have, therefore, $L(0) = p(0) = y_0$. By this we can see that*

$$\prod_{k \in D} (\bar{\alpha}_k)^{\ell_k^0} = \prod_{k \in D} \alpha^{y_k \ell_k^0} = \alpha^{\sum_{k \in D} y_k \ell_k^0} = \alpha^{y_0}$$

as postulated by (VDD.3).

4 Security

The presented distributed key generation protocol follows the variant presented and analyzed in [BNPW24], where it is proven that this protocol provides the expected security guarantees for any threshold $1 \leq t \leq n$ and up to $t-1$ corrupted tellers. More precisely, the proof establishes IND-CPA security of ElGamal encryption using this DKG protocol for key generation, under static corruption of up to $k-1$ tellers.

Note: *The proof in [BNPW24] only requires that zero-knowledge proofs of the knowledge of the discrete logarithm are provided for $A_{k,0}$ ($k \in I$); the ZKPs for $A_{k,i}$ for $i > 0$ are not needed.*

Independently, this version of the DKG protocol has been also analysed in [CL24], where a security proof is given in the UC model (for in ideal functionality defined there). Intuitively, this security result means that all what the participants of the protocol learn are **consistent** shares of s_k $(= a_{k,0})$ chosen by (every) teller k, given as points $p_k(l)$ of a polynomial, along with (consistent) coefficients of this polynomial.

References

[BNPW24] Josh Benaloh, Michael Naehrig, Olivier Pereira, and Dan S. Wallach. ElectionGuard: a cryptographic toolkit to enable verifiable elections. In 33rd USENIX Security Symposium, USENIX Security 2024. USENIX Association, 2024. Available also as `https://eprint.iacr.org/2024/955.`

[CL24] Yi-Hsiu Chen and Yehuda Lindell. Feldman's verifiable secret sharing for a dis honest majority. Cryptology ePrint Archive, Paper 2024/031, 2024. `https://eprint.iacr.org/2024/031.`

[Lagr] `https://en.wikipedia.org/wiki/Lagrange_polynomial`

[TP91] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In Advances in Cryptology - EUROCRYPT 1991, volume 547 of LNCS, pages 522–526. Springer, 1991.