Analysis of Files and Scripts in the "ECDLP-Research" Project

The "ECDLP-Research" project is dedicated to investigating the Elliptic Curve Discrete Logarithm Problem (ECDLP), specifically on the secp256k1 curve, which is widely used in Bitcoin cryptocurrency. The analysis of the provided files (secp256k1.txt, key_list_data_20251002200628.csv, key_list_stats_20251002200628.txt, generate_data.py, and secp256k1.py) elucidates a core aspect of the project: the utilization of the **topological structure of the private key** for key grouping and the potential simplification of private key inversion from the public key. Below is a detailed examination of the purpose of this structure, the grouping mechanism, and the context of its application for inversion using advanced machine learning/deep learning (ML/DL) methods.

1. Purpose of the Topological Structure of the Private Key

The topological structure of the private key represents a detailed record of the sequence of mathematical operations (point doubling and addition on the elliptic curve) performed during the computation of the public key $Q = d \cdot G$, where d is the private key and G is the base point (generator) of the curve. This structure captures the "topology" (or "trajectory") of computations within the curve's point group.

- **Formation Mechanism:**
  - In the file secp256k1.py (Secp256k1 class, scalar_multiply method), the double-and-add algorithm for scalar multiplication (private key) is implemented. During execution, the following steps are logged:
    - Point doubling (operation = ""): $P = 2P$.
    - Point addition (operation = "1", when the private key bit is 1): $Q = Q + P$.
  - The result is a list of rows in the format [x1, y1, x2, y2, x3, y3, operation], where (x1, y1) and (x2, y2) are input points, and (x3, y3) is the result of addition/doubling.
  - The file secp256k1.txt provides an example of such topology for both test and legacy curves. For the test curve, the topology consists of 30 rows, where operation = "1" indicates addition (corresponding to binary 1s in the private key).
- **Relevance to ECDLP:**
  - The ECDLP involves finding d given Q and G. Standard attacks (e.g., Pollard Rho) require computational complexity of $O(\sqrt{n})$, where n is the curve order.
  - The topology reveals the "skeleton" of the binary representation of the private key: the number of additions (total_adds) corresponds to the Hamming weight (number of 1s in binary d), while the positions of doublings/additions reflect the bit structure.
  - The file secp256k1.txt demonstrates inversion of the last point: given the final addition step $(x3, y3) = (x1, y1) + (x2, y2)$, the point $(x1, y1)$ can be recovered as $(x3, y3) - (x2, y2)$ (using the restore_point method in secp256k1.py). This illustrates how topology enables "reverse engineering" of keys, reducing the search space.
- **Sequence of Double Points (double points):**
  - Generated in the get_double_points method: starting from G, the points $2G, 4G, 8G, \ldots$ are computed (up to l=29 for the test curve or l=254 for legacy).

- These points are used for indexing in the topology: for example, the first addition occurs between points with indices idx_x1y1_first and idx_x2y2_first.

2. Grouping of Private Keys by Topological Structure

Grouping is implemented to classify keys based on similar "conditions" extracted from the topology. This segments the vast key space (size $n \approx 2^{256}$ for legacy) into smaller subsets where keys exhibit analogous structures.

- **Condition Determination:**
  - In the identify_condition method (secp256k1.py): a condition is a string in the format "idx_x1y1_first_idx_x2y2_first_total_adds_idx_x2y2_last".
    - idx_x1y1_first: Index of the first point (x1, y1) in the initial addition (operation="1").
    - idx_x2y2_first: Index of the second point (x2, y2) in the initial addition.
    - total_adds: Total number of additions (operation="1").
    - idx_x2y2_last: Index of (x2, y2) in the final addition, where the result is the public key.
  - In secp256k1.txt: For the test curve, the condition is "1_2_16_30", with 27 unknown points, 14 additions, and $C(27, 14) = 20,058,300$ combinations (the count_subsets method computes the binomial coefficient to estimate subset size).
- **Data Generation and Statistics:**
  - The script generate_data.py generates unique private keys (generate_unique_keys method), computes topology for each, extracts the condition, and writes to CSV (key_list_data_*.csv) in the format: condition;private_key;subset_segments (segments are strings of point coordinates for additions).
    - Supports filtering by condition (--filter_condition) and progress display.
  - Statistics in key_list_stats_*.txt: Key counts per condition (using Counter from collections). For example, top conditions: "1_2_15_30" (16,512 keys), "1_2_16_30" (16,255), etc. The top 2,000 conditions cover 96.81% of 1,000,000 keys, with a total of 7,324 unique conditions.
  - In key_list_data_*.csv: Example rows like "2_5_14_30;795978258;..." with segments representing subsets for each condition.
- **Significance of Grouping:**
  - Keys with the same condition share similar binary structures (e.g., comparable Hamming weight and bit positions), reducing entropy within the group.
  - For each condition, the size of unknown point (x2, y2) combination subsets is calculated, making brute-force or optimized attacks feasible within the group.

3. Context of Application for Private Key Inversion

Private key inversion (finding d from Q) is the primary goal of ECDLP. The topological approach transforms the problem into classification and search within subsets:

- **Advantages of the Approach:**
  - Instead of searching the entire space n, keys are grouped by conditions. For the test curve, combinations are small (millions); for legacy, they are large but segmented.

    - Last-point inversion (secp256k1.txt) demonstrates recovery of intermediate points, potentially enabling recursive "unwinding" of the topology back to the private key.
    - Grouping reduces complexity: top conditions cover the majority of keys, allowing focus on them.
- **Potential for ML/DL:**
    - **Classification:** Train a model (e.g., CNN or Transformer) to predict the condition from the public key Q. CSV data serves as an ideal dataset: public key (computed from private) as input, condition as label.
    - **In-Group Inversion:** Within a condition, use DL to predict the binary structure of d (e.g., as a bit sequence). Models such as RNN/LSTM or GAN can generate candidate keys, minimizing a loss function based on distance to Q.
    - **Hybrid Attacks:** Combine with classical methods (Baby-Step Giant-Step) in subsets. ML can optimize subset selection or predict "hot" topology segments.
    - **Scalability:** For the legacy curve (real Bitcoin), generate billions of keys and train on GPU/TPU. Statistics indicate that 2,000 conditions cover ~97%, reducing computations by focusing on them.
    - **Potential Challenges:** High dimensionality (point coordinates ~256 bits), data noise, but topology introduces structure, making the task tractable.

In summary, this approach transforms ECDLP from a "needle in a haystack" problem into a segmented search, where topology serves as the key to space reduction. The project prepares data for ML, enabling future automation of inversion and potentially weakening secp256k1 security for weak keys. If required, extensions to scripts or ML models can be proposed.