WHAT IS CLAIMED IS:

1. A method comprising performing, by a first party computing system:

tokenizing a first party set of data records, thereby generating a tokenized first party set, wherein the tokenized first party set comprises a plurality of first party tokens;

generating a plurality of tokenized first party subsets by assigning each first party token of the plurality of first party tokens to a tokenized first party subset of the plurality of tokenized first party subsets using an assignment function;

assigning each tokenized first party subset to a corresponding first party worker node of a plurality of first party worker nodes, wherein each first party worker node comprises a different first party computing device;

generating a plurality of intersected token subsets by:

for each tokenized first party subset of the plurality of tokenized first party subsets, causing the corresponding first party worker node to:

perform a private set intersection protocol with a corresponding second party worker node of a plurality of second party worker nodes from a second party computing system, thereby privately determining an intersection between the tokenized first party subset and a tokenized second party subset assigned to the corresponding second party worker node, wherein each second party worker node comprises a different second party computing device, the intersection providing an intersected token subset of the plurality of intersected token subsets;

combining the plurality of intersected token subsets, thereby generating an intersected token set; and

detokenizing the intersected token set, thereby generating an intersected set of data records.

2. The method of claim 1, wherein the assignment function comprises a lexicographical ordering function, and wherein generating the plurality of tokenized first party subsets by assigning each first party token of the plurality of first party tokens to one of the plurality of tokenized first party subsets using the assignment function comprises:

assigning each first party token to a corresponding tokenized first party subset based on a lexicographical ordering of the plurality of first party tokens.

3. The method of claim 1, further comprising, for each tokenized first party subset of the plurality of tokenized first party subsets:

determining a padding value, the padding value comprising a difference between a size of the tokenized first party subset and a target value;

generating a plurality of random dummy tokens, the plurality of random dummy tokens comprising a number of random dummy tokens equal to the padding value; and

assigning the plurality of random dummy tokens to the tokenized first party subset.

4. The method of claim 1, wherein the first party computing system comprises a first party server and a first party computing cluster, the first party computing cluster comprising a first party driver node and the plurality of first party worker nodes.

5. The method of claim 1, wherein the first party set corresponds to a set of join keys corresponding to a private database table join query, and wherein the method further comprises:

receiving one or more filtered second party database tables from the second party computing system;

filtering one or more first party database tables using the intersected set, thereby generating one or more filtered first party database tables; and

joining the one or more filtered first party database tables and the one or more filtered second party database tables, thereby generating a joined table.

6. The method of claim 1, further comprising retrieving the first party set from a first party database.

7. The method of claim 1, wherein the first party set comprises a plurality of first party elements, wherein the plurality of first party tokens comprise a plurality of hash values, and wherein tokenizing the first party set comprises generating the plurality of hash values by hashing each first party element using a hash function.

8. The method of claim 1, wherein the intersected set of data records comprises an intersection of the first party set and a second party set corresponding to the

second party computing system, and wherein the first party set and the second party set comprise an equal number of elements.

9. The method of claim 1, wherein the intersected token set comprises a union of the plurality of intersected token subsets, and wherein combining the plurality of intersected token subsets comprises determining the union of the plurality of intersected token subsets.

10. The method of claim 1, further comprising:

prior to tokenizing the first party set:

receiving a request message from an orchestrator computer, the request message indicating the first party set, and

retrieving the first party set from a first party database; and

transmitting, to the orchestrator computer, the intersected set, wherein the orchestrator computer transmits the intersected set to a client computer.

11. The method of claim 1, wherein the plurality of tokenized first party subsets and a plurality of tokenized second party subsets comprise a predetermined number of subsets.

12. The method of claim 11, wherein each tokenized first party subset is associated with a numeric identifier between one and the predetermined number of subsets, inclusive, wherein the assignment function is a hash function that produces hash values between one and the predetermined number of subsets, inclusive, and wherein assigning each first party token of the plurality of first party tokens to one of the plurality of tokenized first party subsets comprises:

for each first party token of the plurality of first party tokens:

generating a hash value using the first party token as an input to the hash function; and

assigning the first party token to a tokenized first party subset with the numeric identifier equal to the hash value.

13. A method comprising performing, by a first party computing system:

receiving a private database table join query, the private database table join query identifying one or more first database tables and one or more attributes;

retrieving the one or more first database tables from a first party database;

determining a plurality of first party join keys based on the one or more first database tables and the one or more attributes;

tokenizing the plurality of first party join keys, thereby generating a tokenized first party join key set, wherein the tokenized first party join key set comprises a plurality of first party tokens;

generating a plurality of tokenized first party subsets by assigning each first party token of the plurality of first party tokens to a tokenized first party subset of the plurality of tokenized first party subsets using an assignment function;

assigning each tokenized first party subset to a corresponding first party worker node of a plurality of first party worker nodes, wherein each first party worker node comprises a different first party computing device;

generating a plurality of intersected token subsets by:

for each tokenized first party subset of the plurality of tokenized first party subsets, causing the corresponding first party worker node to:

perform a private set intersection protocol with a corresponding second party worker node of a plurality of second party worker nodes from a second party computing system, thereby privately determining an intersection between the tokenized first party subset and a tokenized second party subset assigned to the corresponding second party worker node, wherein each second party worker node comprises a different second party computing device, the intersection providing an intersected token subset of the plurality of intersected token subsets;

combining the plurality of intersected token subsets, thereby generating an intersected token set;

detokenizing the intersected token set, thereby generating an intersected join key set;

filtering the one or more first database tables using the intersected join key set, thereby generating one or more filtered first database tables;

receiving, from the second party computing system, one or more filtered second database tables; and

combining the one or more filtered first database tables and the one or more filtered second database tables, thereby generating a joined database table.

14. The method of claim 13, wherein the private database table join query additionally comprises a “where” clause, and wherein the method further comprises, prior to determining the plurality of first party join keys:

pre-filtering the one or more first database tables based on the “where” clause.

15. The method of claim 13, wherein each first party join key of the plurality of first party join keys corresponds to an attribute of the one or more attributes, and wherein tokenizing the plurality of first party join keys comprises:

for each attribute of the one or more attribute, concatenating each first party join key corresponding to the attribute, thereby generating a plurality of concatenated first party join keys; and

hashing each concatenated first party join key of the plurality of concatenated first party join keys, thereby generating a plurality of hash values, wherein the tokenized first party join key set comprises the plurality of hash values.

16. The method of claim 13, further comprising transmitting the one or more filtered first database tables to the second party computing system, wherein the second party computing system combines the one or more filtered first database tables and the one or more filtered second database tables, thereby generating the joined database table.

17. The method of claim 13, wherein the private database table join query is received from an orchestrator computer, and wherein the method further comprises transmitting the joined database table to the orchestrator computer, wherein the orchestrator computer transmits the joined database table to a client computer.

18. The method of claim 13, further comprising:

generating a token column comprising tokenized first party join keys of the tokenized first party join key set; and

appending the token column to the one or more first database tables.

19. The method of claim 18, wherein filtering the one or more first database tables using the intersected join key set comprises:

removing one or more rows from the one or more first database tables based on the token column, the one or more rows corresponding to one or more tokenized first party join keys that are not in the intersected join key set.

20. A first party computing system comprising:

a first processor; and

a first non-transitory computer readable medium coupled to the first processor, the first non-transitory computer readable medium comprising code, executable by the first processor for implementing the method of any of claims 1-19.

21. The first party computing system of claim 20, further comprising a computing cluster comprising a driver node and a plurality of worker nodes, wherein the first processor and the first non-transitory computer readable medium correspond to a first party server, wherein the driver node comprises a second processor and a second non-transitory computer readable medium coupled to the second processor, wherein each worker node of the plurality of worker nodes comprises a third processor of a plurality of third processors and a third non-transitory computer readable medium of a plurality of third non-transitory computer readable mediums, each third non-transitory computer readable medium coupled to a corresponding third processor.