



- (51) International Patent Classification:
H04L 9/00 (2006.01) *H04L 9/14* (2006.01)
- (21) International Application Number:
PCT/US2014/049419
- (22) International Filing Date:
1 August 2014 (01.08.2014)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
61/861,368 1 August 2013 (01.08.2013) US
- (71) Applicant: VISA INTERNATIONAL SERVICE ASSOCIATION [US/US]; P.O. Box 8999, Ms M1-11f, San Francisco, CA 94128-8999 (US).
- (72) Inventors: PAYTON, Paul; 3 Azalea Lane, San Carlos, CA 94070 (US). EDINGTON, Scott; 4500 South Four Mile Run Drive, #115, Arlington, VA 22204 (US). VAN TILBURG, Johan; 12 Waterview Court, Bluffton, SC 29910 (US).
- (74) Agents: BIERNACKI, John, V. et al.; Jones Day, 222 East 41st Street, New York, NY 10017-6702 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— without international search report and to be republished upon receipt of that report (Rule 48.2(g))

(54) Title: HOMOMORPHIC DATABASE OPERATIONS APPARATUSES, METHODS AND SYSTEMS

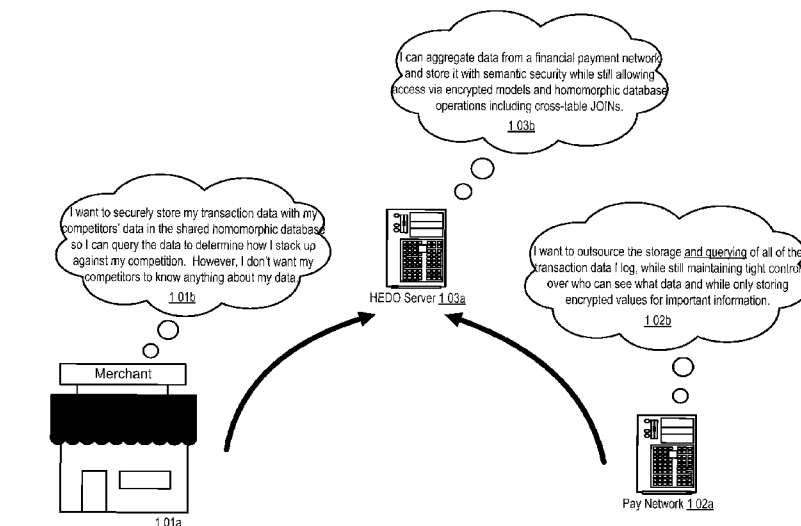


Figure 1

Example: Homomorphically Encrypted Database Operations

(57) Abstract: The homomorphic database operations apparatuses, methods and system ("HEDO") transform transaction storage requests and homomorphic model queries using HEDO components into homomorphic model query results. In some implementations, the disclosure provides a processor-implemented method of securely querying a shared homomorphically encrypted data repository and performing cross-table homomorphic joins.

HOMOMORPHIC DATABASE OPERATIONS APPARATUSES, METHODS AND SYSTEMS

[0001] This application for letters patent disclosure document describes inventive aspects that include various novel innovations (hereinafter “disclosure”) and contains material that is subject to copyright, mask work, and/or other intellectual property protection. The respective owners of such intellectual property have no objection to the facsimile reproduction of the disclosure by anyone as it appears in published Patent Office file/records, but otherwise reserve all rights.

PRIORITY CLAIM

[0002] This application claims priority to United States provisional patent application serial no. 61/861,368, filed August 1, 2013, attorney docket no. 532US01, entitled “Homomorphic Database Operations Apparatuses, Methods and Systems.” The aforementioned application is all hereby expressly incorporated herein by reference.

FIELD

[0003] The present innovations generally address the creation, maintenance, population and querying of secure databases containing full or partial data values stored using somewhat homomorphic, fully homomorphic, or other encryption scheme(s), and more particularly, include HOMOMORPHIC DATABASE OPERATIONS APPARATUSES, METHODS AND SYSTEMS.

[0004] However, in order to develop a reader's understanding of the innovations, disclosures have been compiled into a single description to illustrate and clarify how

1 aspects of these innovations operate independently, interoperate as between individual
2 innovations, and/or cooperate collectively. The application goes on to further describe
3 the interrelations and synergies as between the various innovations; all of which is to
4 further compliance with 35 U.S.C. §112.

5 BACKGROUND

6 **[0005]** Consumers engaging in transactions create large amounts of data, some of
7 which may include private information. For example, a consumer's bank account
8 number, balance, or mailing address may be private information. When making
9 purchases, consumers may choose between multiple merchants offering similar services
10 and in competition with one another. Merchants may desire to optimize the manner in
11 which they attract and/or retain customers.

12 BRIEF DESCRIPTION OF THE DRAWINGS

13 **[0006]** The accompanying appendices and/or drawings illustrate various non-
14 limiting, example, innovative aspects in accordance with the present descriptions:

15 **[0007]** FIGURE 1 shows an example block diagram illustrating aspects of the
16 HEDO, in one implementation of the HEDO operation;

17 **[0008]** FIGURE 2 shows an example data flow illustrating aspects of
18 homomorphically encrypted database transaction data aggregation, in one
19 implementation of the HEDO operation;

1 **[0009]** FIGURE 3 shows an example data flow illustrating aspects of querying and
2 joining homomorphically encrypted databases, in one implementation of the HEDO
3 operation;

4 **[0010]** FIGURES 4A-B show an example logic flow illustrating aspects of
5 performing homomorphic joins, e.g., an example HJ Component, in one
6 implementation of the HEDO operation;

7 **[0011]** FIGURE 5 shows an example logic flow illustrating aspects of
8 homomorphic calculations in database space, e.g., an example HOP Component, in one
9 implementation of the HEDO operation;

10 **[0012]** FIGURE 6 shows an example logic flow illustrating aspects of generating
11 HEDO packages, e.g., an example GHP Component, in one implementation of the
12 HEDO operation;

13 **[0013]** FIGURES 7A-G show example user interfaces illustrating aspects of the
14 HEDO, in one implementation of the HEDO operation; and

15 **[0014]** FIGURE 8 shows a block diagram illustrating aspects of an exemplary
16 embodiment of a HEDO user interface controller, in one implementation of the HEDO
17 operation.

18 **[0015]** FIGURES 9-12 illustrate further aspects of the HEDO, in one
19 implementation of the HEDO operation.

20 **[0016]** The leading number of each reference number within the drawings
21 indicates the figure in which that reference number is introduced and/or detailed. As

¹ such, a detailed discussion of reference number 101 would be found and/or introduced
² in Figure 1. Reference number 201 is introduced in Figure 2, etc.

³

DETAILED DESCRIPTION

HEDO

[0017] The HOMOMORPHIC DATABASE OPERATIONS APPARATUSES, METHODS AND SYSTEMS (hereinafter “HEDO” user interface) transform transaction data aggregation and homomorphic database model queries, via HEDO components, in response to user activities. In some embodiments, this is carried out in real time.

[0018] Fig. 1 shows an example block diagram illustrating aspects of the HEDO, in one implementation of the HEDO operation. In some embodiments, a merchant 101a may wish to store their transaction data in a shared homomorphic database along with their competitors’ data, e.g., 101b. In so doing, the merchant can query the data to determine how they compare against competitors using a variety of comparison metrics without having access to their competitors’ underlying data. Similarly, competitors can perform such queries without having access to the merchant’s specific data. In one embodiment, a pay network server 102a may wish to outsource the storage and the querying of transaction data and may forward data that the server receives from merchants and consumers engaging in transactions, e.g., 102b. However, the pay network server may wish to maintain tight controls over the data such as limiting who can see the data stored as well as having assurances that important data point values remain encrypted when stored using the outsourced solution. In one embodiment, HEDO server 103a may aggregate data from a financial payment network and store it such that the data is encrypted and has semantic security. Semantic security in this context meaning that an adversary with possession of the stored data would be unable to

1 determine substantial properties or values of the stored data. In some embodiments,
2 the HEDO server 103a a may allow access to the stored encrypted data and may perform
3 operations to evaluate encrypted models through the use of homomorphic operations
4 including homomorphic cross-table joins, e.g., 103b.

5 **[0019]** Fig. 2 shows an example data flow illustrating aspects of homomorphically
6 encrypted database transaction data aggregation, in one implementation of the HEDO
7 operation. In one embodiment, a transaction input 202a may be forwarded to merchant
8 transaction server 201a, 201b. The merchant transaction server may process the
9 transaction and initiate a transaction storage request 203a,203b to HEDO client 204. A
10 transaction storage request may be a request to encrypt certain values in the transaction
11 storage request while leaving other values unencrypted, and facilitate the insertion of
12 those values and that record into an HEDO repository server. In one embodiment, an
13 HEDO package may be generated 205 upon the processing of the transaction storage
14 request. Further detail with regard to HEDO package generation may be found with
15 respect to Fig. 6, e.g. a GHP component 600. In one embodiment, grouping keys that
16 may be utilized by the HEDO client, the HEDO server, and/or the HEDO repository
17 server at a future point may be generated during the creation of the HEDO package. In
18 one embodiment, the grouping keys are inserted 206 into an HEDO client database 207.
19 In other embodiments, the grouping keys may be retransmitted back to the merchant
20 transaction server, forwarded to the HEDO server, stored in the HEDO repository,
21 and/or the like. In one embodiment, the HEDO client may then transmit an HEDO
22 package storage request 208 to HEDO server 209. The HEDO server may then store the
23 package contents in a repository. An example HEDO repository server schema suitable

1 for storing merchant transaction server transaction storage requests, substantially in the
2 form of SQL statements is:

```
3 CREATE TABLE IssuerE
4 (Number VARCHAR(16) PRIMARY KEY
5 );
6
7 CREATE TABLE MerchantE
8 (Number VARCHAR(16) PRIMARY KEY,
9 Category CHAR(4) NOT NULL
10 );
11
12 CREATE TABLE CardholderE
13 (Number VARCHAR(250000) PRIMARY KEY, --HOMOMORPHICALLY ENCRYPTED
14 Surname VARCHAR(64) NOT NULL,
15 Issuer VARCHAR(16) NOT NULL,
16 Product VARCHAR(3) NOT NULL,
17 Risk VARCHAR(50000) NOT NULL --HOMOMORPHICALLY ENCRYPTED
18 );
19
20 CREATE TABLE TransactionE
21 (Number VARCHAR(64) PRIMARY KEY,
22 Merchant VARCHAR(16) NOT NULL,
23 Cardholder VARCHAR(250000) NOT NULL, --HOMOMORPHICALLY ENCRYPTED
24 Type VARCHAR(3) NOT NULL,
25 Amount VARCHAR(250000) NOT NULL --HOMOMORPHICALLY ENCRYPTED
26 );
27
28 CREATE TABLE PurchaseE
29 (Transaction VARCHAR(64) NOT NULL,
30 Surname VARCHAR(2048000) NOT NULL, --HOMOMORPHICALLY ENCRYPTED
31 Merchant VARCHAR(16) NOT NULL,
32 Product VARCHAR(32) NOT NULL,
33 Quantity INTEGER NOT NULL
34 );
35
```

```
1  CREATE TABLE ProductE
2  (Code VARCHAR(32) PRIMARY KEY,
3   Good VARCHAR(256) NOT NULL,
4   Price DECIMAL(6,2) NOT NULL
5  );
```

6 **[0020]** An example alternative HEDO repository server schema suitable for
7 storing merchant transaction server transaction storage requests and maintaining
8 grouping values for use with grouping keys, substantially in the form of SQL statements
9 is:

```
10 CREATE TABLE IssuerE
11 (Number VARCHAR(16) PRIMARY KEY
12 );
13
14 CREATE TABLE MerchantE
15 (Number VARCHAR(16) PRIMARY KEY,
16  Category CHAR(4) NOT NULL
17 );
18
19 CREATE TABLE CardholderE
20 (Number VARCHAR(250000) PRIMARY KEY,
21  Number_grp VARCHAR(250000) NOT NULL,
22  Number_gkey VARCHAR(250000) NOT NULL,
23  Surname VARCHAR(64) NOT NULL,
24  Issuer VARCHAR(16) NOT NULL,
25  Product VARCHAR(3) NOT NULL,
26  Risk VARCHAR(50000) NOT NULL,
27  Risk_grp VARCHAR(50000) NOT NULL,
28  Risk_gkey VARCHAR(50000) NOT NULL
29 );
30
31 CREATE TABLE TransactionE
32 (Number VARCHAR(64) PRIMARY KEY,
33  Merchant VARCHAR(16) NOT NULL,
```

```
1   Cardholder VARCHAR(250000) NOT NULL,
2   Cardholder_grp VARCHAR(250000) NOT NULL,
3   Cardholder_gkey VARCHAR(250000) NOT NULL,
4   Type VARCHAR(3) NOT NULL,
5   Amount VARCHAR(250000) NOT NULL,
6   Amount_grp VARCHAR(250000) NOT NULL,
7   Amount_gkey VARCHAR(250000) NOT NULL
8 );
9
10  CREATE TABLE PurchaseE
11  (Transaction VARCHAR(64) NOT NULL,
12   Surname VARCHAR(2048000) NOT NULL,
13   Surname_grp VARCHAR(2048000) NOT NULL,
14   Surname_gkey VARCHAR(2048000) NOT NULL,
15   Merchant VARCHAR(16) NOT NULL,
16   Product VARCHAR(32) NOT NULL,
17   Quantity INTEGER NOT NULL
18 );
19
20  CREATE TABLE ProductE
21  (Code VARCHAR(32) PRIMARY KEY,
22   Good VARCHAR(256) NOT NULL,
23   Price DECIMAL(6,2) NOT NULL
24 );
```

25 **[0021]** In some embodiments, the repository is an open source or commercial
26 database running and in communication with the HEDO server 209. Databases suitable
27 for this purpose include but are not limited to PostgreSQL, MySQL, Oracle, and/or the
28 like. In other embodiments, the HEDO repository may be contained on a different
29 server such as the HEDO repository server 211, and/or the HEDO client 204 itself. In
30 one embodiment, homomorphically encrypted values, homomorphic encrypted
31 grouping values, and/or unencrypted data values, e.g., 210, may be forwarded from the
32 HEDO server 209 to the HEDO repository server 211. The HEDO repository server 211

1 may then perform an insertion into the repository of the encrypted and non-encrypted
 2 values, 212. In some embodiments, the HEDO repository server may be a database
 3 server that is configured with user-defined functions to further process the encrypted
 4 and/or unencrypted values contained in homomorphic insertion 212. For example, in
 5 one embodiment, all of the data forwarded to the HEDO repository server 211 may be in
 6 unencrypted form. In such embodiments, a user defined function, stored procedure,
 7 and/or the like running and in communication with HEDO repository server 211 may
 8 selectively or completely encrypt the values from homomorphic insertion 212 using
 9 either deterministic or non-deterministic homomorphic encryption. An example user
 10 defined function suitable for encrypting a binary decimal value (such as a digit of a
 11 credit card number), substantially in the form of Python executable code is:

```

12 def BCD_Encrypt(Number, Lambda, p) :
13     number = abs(Number) % 10 # ensure 0-9 [no A-F!]
14     Eight = ((Number & 0x08) != 0)
15     Four  = ((Number & 0x04) != 0)
16     Two   = ((Number & 0x02) != 0)
17     One   = ((Number & 0x01) != 0)
18
19     BCD = [Encrypt(One , Lambda, p) , \
20           Encrypt(Two  , Lambda, p) , \
21           Encrypt(Four , Lambda, p) , \
22           Encrypt(Eight, Lambda, p) ]
23     return BCD

```

24 **[0022]** In other embodiments, more complicated validation may be performed on
 25 the data before or after encryption. An example of such a validation may be the LUHN
 26 algorithm used for validating credit card numbers. An example implementation of the

1 LUHN algorithm, suitable for use on homomorphically encrypted HEDO data,
2 substantially in the form of Python code is:

```

3  def LuhnAlgorithm(BCD, PAN_Length):
4      Zero = cZERO; Tally0 = BCD_Zero; Tally1 = BCD_Zero; Tally2 =
5  BCD_Zero
6
7      # Double the BCD values of selected digits
8      # starting at the check digit.
9      for Position in range(PAN_Length-2,-1,-2):
10         Digit = BCD[Position]
11         [Digit0,Digit1] = BCD_Double(Digit)
12         Digit = BCD_Resolve(Digit0,Digit1)
13         BCD[Position] = Digit
14
15         # Sum all the resulting digits, producing
16         # a tally of their values.
17         for Position in range(PAN_Length):
18             Digit = BCD[Position]
19             Answer = BCD_Sum(Tally0,Tally1,Tally2,Digit)
20             Tally0 = Answer[0]; Tally1 = Answer[1]; Tally2 =
21 Answer[2]
22
23         # Return the least significant BCD digit
24         # (modulo ten operation).
25         return Tally0

```

26 **[0023]** Fig. 3 shows an example data flow illustrating aspects of querying and
27 joining homomorphically encrypted databases, in one implementation of the HEDO
28 operation. In one embodiment, user 301 may initiate a model query input 302. A
29 model query input may consist of a query or query parameters that are suitable for
30 querying an HEDO repository. In one embodiment, suitable query models may be
31 stored on HEDO server 310, and called by name by user 301 in a model query input 302.

1 In one embodiment, HEDO server 310 may contain a parameterized query which the
2 model query input may invoke by name. For example, a query "loyalty_comparison"
3 stored on or in communication with the HEDO server may be invoked through a model
4 query referencing the query by name and providing suitable parameters such as a
5 merchant identifier, issuer identifier, and/or the like. Example model query requests
6 may be found herein and particularly with respect to Fig. 7.

7 **[0024]** In one embodiment, the model query request 303 is forwarded to HEDO
8 client 304. The HEDO client may convert the model query to HEDO queries, may
9 determine the join, order, and/or grouped by tables required to execute the query,
10 and/or retrieve grouping keys from a database in communication with the HEDO client,
11 e.g., 305. In one embodiment, the HEDO client 304 may request grouping keys from
12 HEDO client database 307, e.g., request grouping keys 306. The database may retrieve
13 keys associated with tables of the query that may require aggregation, cross-table
14 joining, and/or the like. In one embodiment, the HEDO client may forward an HEDO
15 model query request containing HEDO queries and, optionally, required grouping keys,
16 e.g. 309 to HEDO server 310. An example HEDO model query request, substantially in
17 the form of an HTTP(S) POST message including XML-formatted data, is provided
18 below:

```
19 POST /hedo_model_query_request.php HTTP/1.1
20 Host: www.hedoserver.com
21 Content-Type: Application/XML
22 Content-Length: 667
23 <?XML version = "1.0" encoding = "UTF-8"?>
24 <hedo_model_query_request>
25   <timestamp>2020-12-12 15:22:43</timestamp>
```

```

1 <user_name>HEDO_CLIENT</user_name>
2 <credentials>
3   <password>secretpass1234</password>
4   <private_key>h767kwjiwnfe456#@hnniimidrtsxbi</private_key>
5 </credentials>
6 <model_query name="loyalty_comparison">
7   <query num=1>
8     select MerchantE.Category into MCC from
9     MerchantE where MerchantE.Number = '{Merchant}'
10  </query>
11  <query num=2>
12    select CardholderE.Surname, TransactionE.Amount
13    from TransactionE, CardholderE where
14    CardholderE.Number = TransactionE.Cardholder
15    and TransactionE.Merchant = '{Merchant}'
16  </query>
17  <query>
18    ...
19  </query>
20 </model_query>
21 <grouping_keys>
22   <group name="CardholderE.Number = TransactionE.Cardholder">
23     <table1 name="CardholderE" field="Number">
24       <key value="id:1-2000">
25         Jukyfttrddesrsdxerr43ser43wshryt
26         Erwsadxtyjfgkftyresagt43esy45wuykg
27         Sre4fytkuhopj;/945ur6tsgred-yukg
28       </key>
29       <key value="id:rest">
30         ...
31       </key>
32       <key default="true">
33         ...
34       </key>
35       <key>
36         ...
37       </key>

```

```
1         </table1>
2         <tableN>
3             ...
4         </tableN>
5     </group>
6 </grouping_keys>
7 </hedo_model_query_request>
```

8 **[0025]** In some embodiments, HEDO server 310 may receive the model query
9 request and extract individual HEDO query operations, e.g., 311. For example, in one
10 embodiment, an HEDO model query may contain multiple queries. An example first
11 query may include a query that determines a merchant's customer loyalty score. In such
12 an embodiment, a second query may then determine the loyalty scores of the merchant's
13 customers with respect to other merchants in competition with the querying merchant.
14 In so doing, the HEDO server may execute multiple discrete model queries and,
15 optionally, integrate the resultant values either through SQL aggregation options, user-
16 specified function processing, and/or the like. In one embodiment, a first HEDO query
17 312 is forwarded to the HEDO repository 313. As discussed herein, in some
18 embodiments, the HEDO repository is in fact a database in communication with HEDO
19 server 310. In other embodiments, the repository is stored on a third-party cloud
20 provider platform such as Amazon AWS™, RackSpace Cloud™, and/or the like.

21 **[0026]** In one embodiment, the HEDO repository may determine that multiple
22 tables are required to be queried in order to successfully process the query, and may
23 further determine that table values that are to be joined on are in fact homomorphically
24 encrypted values. The HEDO repository may process the HEDO query and perform the
25 required homomorphic joins, e.g. 314. Further detail regarding performing

1 homomorphic joins may be found with respect to Figs. 4A-B, e.g., an example HJ
2 Component 400. In some embodiments, the homomorphically joined tables may
3 require additional homomorphic calculations to be performed on the results set in order
4 to complete the HEDO query execution, e.g., 315. For example, the homomorphically
5 joined results set may require summation. However, because the values are
6 homomorphically encrypted, traditional or default summation interfaces integrated into
7 the database software may not be suitable for performing summations on large or
8 complex data values such as may be found with respect to homomorphic encryption. As
9 such, specialized homomorphic operations may be performed on homomorphic data
10 instead of the normal equivalent database operations. In other embodiments, the
11 HEDO repository may determine that the interfaces integrated into the database
12 software are suitable for performing calculations on homomorphically encrypted data.
13 Periodically, the HEDO repository may re-encrypt, truncate, and/or reset the length of a
14 final or interim homomorphic calculation value in order to maintain the ability for the
15 HEDO repository and/or an underlying database implementation to be able to maintain
16 the values and usable form. Further detail with regard to performing homomorphic
17 calculations, re-encrypting, truncating, and/or resetting the length of homomorphic
18 calculation outputs (e.g., "Homomorphic garbage collection"), may be found with
19 respect to Fig. 5, e.g., an example HOP component 500. In one embodiment, upon the
20 completion of the homomorphic calculations on the HEDO repository results set,
21 aggregation, grouping and/or ordering of the return values may be required. These
22 operations may, in one embodiment, be performed on homomorphically encrypted data.
23 As such, facilities such as those described with respect to homomorphic joins may be
24 employed in order to facilitate the grouping, ordering, and/or other aggregation of the

1 results set. In one embodiment, wherein the aggregation column returned from the
2 HEDO repository has been encrypted using a key that is known or obtainable, the
3 HEDO repository may use the key to decrypt and group/order on a homomorphically
4 encrypted field, e.g., 316. In one embodiment, the HEDO repository may return the
5 results, e.g., HEDO results response 317. An example HEDO results response,
6 substantially in the form of an HTTP(S) POST message including XML-formatted data,
7 is provided below:

```
8   POST /hedo_results_response.php HTTP/1.1
9   Host: www.hedoserver.com
10  Content-Type: Application/XML
11  Content-Length: 667
12  <?XML version = "1.0" encoding = "UTF-8"?>
13  <hedo_results_response>
14    <timestamp>2020-12-12 15:23:18</timestamp>
15    <user_name>HEDO_REPOSITORY_SERVER</user_name>
16    <credentials>
17      <password>secretpass5678</password>
18      <private_key>TGDFRRED456#@hnniimidrtsxabi</private_key>
19    </credentials>
20    <query_results name="loyalty_comparison">
21      <component val="1" name="merchant_customer_loyalty">
22        <res type="homomorphic_encrypted">
23          {merchant}'s customers are {crypt_loyal_val} loyal
24        </res>
25        <val name="crypt_loyal_val"
26          decryptable_by="client_server_only"
27          HEDO_server_can_decrypt="false">
28          111010001000101111111000001111
29          0001110111100011010101111101111
30        </val>
31      </component>
32      <component val="2" name="customer_market_loyalty">
```

```
1      <res type="cleartext_result" cmloyal="78%">
2          Merchant {merchant}'s customers are {cmloyal}
3          more loyal to competitors among the group Gap
4          and Nordstrom
5      </res>
6  </component>
7  </query_results>
8  <hedo_results_response>
```

9 **[0027]** The HEDO server may perform post-processing of the query results
10 and/or supplemental aggregation 318. It should be noted that alternative
11 implementations of the HEDO may perform homomorphic joining, homomorphic
12 aggregation, homomorphic grouping, and/or the like at any point in the data path. In so
13 doing, and HEDO may be configured such that more or less computation may be
14 outsourced to a third-party less-trusted server depending on varying levels of user risk
15 tolerance. An example implementation wherein HEDO server 310 performs
16 homomorphic aggregation/grouping on a result set returned from HEDO repository
17 313, written substantially in the form of executable Python code is:

```
18 def Query(Parameters):
19     #Execute HEDO query
20     SQL_Command = Query1A.format(**Parameters)
21     Cursor.execute(SQL_Command)
22     Rows = Cursor.fetchall()
23     Results = list(); MyResults = list()
24     for Row in Rows:
25         Risk = Encryption.ASCII_Decrypt(eval(Row[0]),Encryption.p)
26         Count = Row[1]
27         Results.append(list([Risk,Count]))
28     print 'A total of',len(Results),'rows were returned.'
29
30     # Perform HEDO aggregation in memory on server
```

```

1      RiskValues = sorted(map(lambda x: x[0], Results))
2      RiskValues = [k for k,g in itertools.groupby(RiskValues)]
3      for Risk in RiskValues:
4          Members = [Item for Item in Results if
5                      Item[0] == Risk]; Count = 0
6          for Member in Members:
7              Count += Member[1]
8          print 'Risk level',Risk, 'had',Count, 'rows.'
9          Count = Encryption.Binary_Encrypt
10                 (Count,Encryption.Lambda,Encryption.p)
11      MyResults.append
12                 (dict(zip(Query1Fields,list([Risk,Count]))))
13      print

```

14 **[0028]** In one embodiment, HEDO server 310 may return the aggregated results
15 to HEDO client 304, e.g., HEDO model query results 319. As discussed above, the
16 HEDO client may perform supplemental client-side HEDO result processing 320. For
17 example, in an HEDO configuration wherein the keys required to properly aggregate
18 and/or group result data are unavailable to the HEDO server, the HEDO client may
19 perform the role of aggregator while the HEDO server and/or HEDO repository
20 performs the role of querying. Such a configuration may be beneficial in the case where
21 an HEDO client is a trusted server and the HEDO server and/or HEDO repository are
22 suspect or less-trusted servers. In one embodiment, HEDO client 304 will return the
23 query results, e.g., model query response 321 and the user client device may render the
24 model query output, e.g., 322.

25 **[0029]** FIGURES 4A-B show an example logic flow illustrating aspects of
26 performing homomorphic joins, e.g., an example HJ Component, in one
27 implementation of the HEDO operation. In one embodiment, and HEDO query 403

1 may be received from HEDO server 401. The HEDO repository 402, which may itself be
2 a server or a database in communication with the HEDO server, may process the HEDO
3 query using a HEDO lexer. A lexical analyzer such as Lex and/or other suitable
4 commercial products may be used for this purpose. In one embodiment, if a query is
5 determined not to span multiple tables 405, a signal may be sent to the default database
6 query executor to handle the non-joined query 406. In other embodiments, if the query
7 is determined to span multiple tables, the lex output tokens may be searched for a
8 joining keyword 407. A joining keyword may be any standard or nonstandard SQL
9 keyword that designates a joined query. Examples of joining keywords include INNER,
10 OUTER, and/or the like. If a joining keyword is not found 408, the HEDO repository
11 may in one embodiment default the query to an INNER JOIN 409. In other
12 embodiments, if a joining keyword is found 408 the type of joining may be determined
13 based on the joining keyword token 410. For example, a LEFT join keyword may
14 instruct the HEDO repository to later select a particular homomorphic joins strategy
15 based on the joined type. In one embodiment, the tokens may be examined to
16 determine the tables that are joined in the query 411. Furthermore, the relevant tokens
17 may be further processed or parsed to determine the fields upon which the tables should
18 be joined 412. For example, in one embodiment, the text "foo.bar" may signify that the
19 "bar" field on table "foo" is to be joined. In one embodiment, the HEDO repository
20 and/or HEDO server may determine if the tables are to be joined on a homomorphic
21 Lee encrypted value. Homomorphic Lee encrypted fields may have certain
22 characteristics that make JOIN strategies employed by open source and commercial
23 database systems ineffective to join tables containing homomorphic Lee encrypted
24 values. For example, values that have had homomorphic operations performed upon

1 them may grow exponentially in the operations performed. As more and more storage
2 is required to encapsulate the record value, joins of homomorphic values may swamp
3 built in joined query executions. Therefore, if the join is not a homomorphic join 414,
4 the join may be passed to the default database query executor such that that query
5 executor may handle the non-homomorphic join 406. In other embodiments, if the join
6 is a homomorphic join 414, the HEDO repository 402 may determine a homomorphic
7 join strategy based on the join column attributes 415. The join column attributes may
8 be attributes that are inherent to the field type of the joined column, such as extremely
9 large variable character fields or large integer fields used in contexts where smaller
10 storage requirements would be expected. The join strategy may also be determined
11 based on the join column names, comments associated with the join columns and stored
12 with the database table, and/or the like. In some embodiments, the homomorphically
13 encrypted field values may have been encrypted using a deterministic homomorphic
14 strategy. As such, when a given value such as a consumer's credit card number is
15 deterministically homomorphically encrypted, there can be assurances that future
16 encryptions of the same value will result in the same ciphertext. Under this
17 homomorphic join strategy record values for homomorphically encrypted data may be
18 directly compared across database tables as the inherent variability and noise associated
19 with various encryption schemes may have been minimized through the use of a
20 deterministic homomorphic encryption scheme. In one embodiment, if the tables can
21 be homomorphically joined directly on the record values 416, then the join may be
22 performed using the appropriate JOIN ON field as though the field were a non-
23 encrypted value 421. In other embodiments, if the tables may not be homomorphically
24 joined directly on the record values 416, it may be determined whether the key that was

1 used to homomorphically encrypt the record values is available to the HEDO repository
2 402, e.g., 417. For example, in one embodiment, the HEDO repository may query an
3 HEDO client for an encryption key to use in performing the requested join. In other
4 embodiments, the HEDO repository and/or the HEDO server may directly store the
5 required join key or other structure such as a garbled circuit that may be used to
6 perform the join. In one embodiment, the HEDO repository 402 may retrieve the key
7 used to encrypt JOIN ON values in each JOIN table 422. The retrieved key may then be
8 used to decrypt the required JOIN ON values 423. In one embodiment, the join may
9 now be performed using the decrypted JOIN ON values, e.g. 421. In still other
10 embodiments, if the key that was used to encrypt homomorphically the joined record
11 values is not available 417, the HEDO query may be analyzed to determine if it contains
12 one or more grouping keys 418. If no grouping keys are available a non-parsable query
13 exception may be generated 419 and returned to the HEDO server 401. The HEDO
14 server 401 may in turn output a non-parsable query error. In other embodiments, if the
15 HEDO query does contain grouping keys 418 the keys may be extracted from the HEDO
16 query 421 and a grouping field associated with each column required for the join may be
17 determined 425. Grouping fields may, in one embodiment, be additional fields that are
18 either added directly to the database table schema or maintained in a hidden table or
19 other structure suitable for retrieval, such that while the value of the associated
20 encrypted field may not be available and no information regarding the similarity of the
21 unencrypted values may be determined as a nondeterministic homomorphic encryption
22 scheme may have been used, the grouping field may be utilized to enable a database join
23 to be completed. In some embodiments, the grouping fields may contain values that
24 have been stored as a result of a deterministic encryption scheme. In such

1 embodiments, the penetration or decryption of the encrypted grouping field values
2 would only compromise the underlying value used for grouping or joins. In still other
3 embodiments, the grouping fields may be encrypted using a nondeterministic
4 encryption scheme such that one or more grouping keys may be required in order to
5 determine the underlying grouping field value and therefore in order to perform a
6 successful join. In one embodiment, if only one grouping keys present 426, that
7 grouping key may be assumed to apply to all grouping field values and as such may be
8 used to decrypt the record values for each JOIN ON grouping field using the singular
9 key 428. In other embodiments, if more than one grouping keys present 426, the HEDO
10 repository 402 may determine whether the plurality of grouping keys may be correlated
11 with record ranges in the grouping field that each key correspondingly encrypts 427. In
12 one embodiment, the plurality of keys may be used to decrypt the grouping field values
13 in total by a applying the decryption keys to the various ranges to which they apply 428.
14 In still other embodiments, if the plurality of grouping keys cannot be definitively
15 associated with the range of grouping field values that they encrypt 427, for each key
16 within the plurality of grouping keys the HEDO repository may decrypt the record
17 values for each JOIN ON grouping field using the cake 429. In one embodiment, the
18 decrypted output values for a given grouping field value may be examined to determine
19 the value that is what in the grouping field acceptable value range and that value may
20 then be used as the decrypted grouping field value 430. In further embodiments, upon
21 completion of the decryption, a join may be performed using the decrypted grouping
22 field values associated with the JOIN ON field in each table 421.

1 **[0030]** Fig. 5 shows an example logic flow illustrating aspects of homomorphic
2 calculations in database space, e.g., an example HOP Component, in one
3 implementation of the HEDO operation. In one embodiment, the HEDO repository 501
4 may analyze a joined or non-joined query to determine which columns or records in the
5 result set require further processing. Further processing may include grouping,
6 aggregation, summation, user-specified function processing, and/or the like 502. In one
7 embodiment, the processing needs may be used to determine which underlying
8 database operations are required to successfully process the query completely 503. For
9 example, field values may be required to be added in order to determine a sum value of
10 all records returned. Similarly, result set field values may be required to be compared
11 against each other so as to determine the minimum value within the result set.
12 However, in some embodiments, the presence of homomorphically encrypted data may
13 frustrate the execution of such database operations. In one embodiment, if the column,
14 record, and/or value to be operated upon in the result set is not homomorphically
15 encrypted the HEDO repository may invoke the normal underlying database operation
16 on the value 505. In other embodiments, if the value is homomorphically encrypted 504
17 the HEDO repository may query a homomorphic library for a homomorphic operation
18 that may be substituted for the database operation. For example, in the case where
19 multiple values in a result set must be added in order to determine a final value, the
20 homomorphic library may contain a homomorphically optimized addition function to
21 facilitate the adding of homomorphically encrypted values 506. In other embodiments,
22 the homomorphic operation is the same as the default database operation. If a
23 homomorphic operation is not found, in one embodiment, the default database
24 operation may still be applied to the data values. In so doing, operations that in their

1 normal usage would work equally well on homomorphically encrypted values may be
2 utilized without modifications for homomorphic encryption. In other embodiments, if a
3 homomorphic operation is found 507, the homomorphic operation may be executed on
4 a portion or all of the data values 508. In some embodiments, the smallest feasible
5 portion of a data value may be used to perform a homomorphic operation. For example,
6 one implementation of a non-homomorphic mean value determination function may
7 sum all of the result values and then divide the total by the number of values
8 encountered. In a homomorphic version of mean value determination, however, it may
9 be more computationally beneficial to incrementally calculate the mean as each
10 additional record value is added. In one embodiment, the required storage attributes
11 for a column or a memory location that may hold the resultant value from the
12 homomorphic operation may be determined 509. If the value resulting from the
13 homomorphic operation exceeds the required storage attributes 510, homomorphic
14 garbage collection may be performed on the resultant value 511. Homomorphic garbage
15 collection may, in some embodiments, be used to shorten or truncate the length of the
16 value produced by a homomorphic operation. In some embodiments, the garbage
17 collection may take the form of a full decryption followed by a full encryption. In other
18 embodiments, the garbage collection may take the form of a re-crypt operation, wherein
19 the actual unencrypted value of the result of the homomorphic operation is never
20 calculated by the garbage collection function. In one embodiment, if more incremental
21 homomorphic operation processing is required 512, the next feasible portion of the data
22 value may be homomorphically evaluated along with the previous value, e.g., 508.

1 **[0031]** Fig. 6 shows an example logic flow illustrating aspects of generating
2 HEDO packages, e.g., an example GHP Component, in one implementation of the
3 HEDO operation. In one embodiment, merchant transaction server 601 may transmit a
4 transaction storage request 604 to HEDO client 602. In one embodiment, the HEDO
5 client may extract values from the transaction storage request and normalize the data,
6 e.g., 605. Normalizing may include trimming strings, casting values two different data
7 types, querying a data schema table to determine required properties of a transaction
8 storage request value, and/or the like. In one embodiment, the HEDO client may query
9 for a storage schema template 606. The HEDO database 603 may retrieve the template
10 and return an applicable storage schema 607. In one embodiment, the HEDO client
11 may then match the schema metadata values to the normalized data values extracted
12 from the transaction storage request, e.g., 608. In one embodiment, all of the values are
13 marked as unprocessed and the first unprocessed value is then extracted 609. If the
14 unprocessed value does not need to be homomorphically encrypted 610, the HEDO
15 client may determine whether there are more unprocessed values 611. If there are no
16 unprocessed values the procedure may exit. If there are unprocessed values the next
17 unprocessed value may be extracted. If the value under examination does need to be
18 homomorphically encrypted 610, the HEDO client may determine whether a
19 deterministic homomorphic encryption is available 612. If a deterministic
20 homomorphic encryption is available, the value may be encrypted using such a scheme
21 613. If deterministic homomorphic encryption is not available, the HEDO client may
22 determine whether a nondeterministic homomorphic encryption is available 615. If no
23 such nondeterministic homomorphic encryption is available the client may return a
24 storage encryption error 616. If a nondeterministic homomorphic encryption is

1 available 615, the HEDO client may determine if there is a unitary key used for
2 nondeterministic encryption. If a unitary key is used the HEDO client will load the key
3 618 and encrypt the values using the unitary key 619. If the unitary key is not used for
4 nondeterministic homomorphic encryption 617, the HEDO client may determine if a
5 previously created grouping key can be used to encrypt the extracted values 620. If a
6 previously used grouping key is not available 621, a new grouping key may be created
7 622. In some embodiments, the grouping key may be forwarded to an HEDO client
8 database for storage. In other embodiments, the grouping key may be stored by the
9 HEDO server, the HEDO repository, the merchant transaction server, and/or the like.
10 In still other embodiments, the grouping key may be supplemented with metadata such
11 as but not limited to, the table, the fields, and/or the range of values that the grouping
12 key encrypts. In so doing, the HEDO may later utilize the grouping key in performing
13 homomorphic joins, homomorphic aggregation, and/or other homomorphic operations.
14 Once a grouping key is obtained, in some embodiments, the extracted values may be
15 encrypted using the grouping key 623. If more values are unprocessed 614, the
16 procedure may repeat, otherwise the procedure may terminate.

17 **[0032]** FIGURES 7A-G show example user interfaces illustrating aspects of the
18 HEDO, in one implementation of the HEDO operation. In one embodiment, the HEDO
19 user interface 701a, allows merchants to easily perform useful queries against the HEDO
20 database without having access to the underlying data values. In one embodiment, a
21 merchant may desire to detect fraud, e.g. 701b, and may define fraud as multiple
22 purchases made by the same cardholder at the same merchant type with cards from
23 multiple issuers. In other embodiments, the merchant may desire to compare its known

1 customers' transactions to those transactions its customers make at the merchants
2 competitors, 701c. In still other embodiments, the merchant may desire to know how
3 the risk levels associated with its customers compare to those of the customers of its
4 competitors, 701d.

5 **[0033]** With respect to Fig. 7B, and HEDO user interface, e.g. 702a, may be
6 presented for fraud detection. In one embodiment, competitors of the merchant may be
7 displayed 702b as well as a SQL-like query 702c that allows a merchant to specify an
8 issuer for which to determine levels of fraud. In other embodiments, other entities such
9 as the issuer themselves, a consumer and/or the like may utilize the query facilities. In
10 one embodiment, the results of the query are displayed 702d and show instances where
11 customers utilize different cards to engage in transactions with merchants of the same
12 type. In still other embodiments, a table may be displayed containing the customer's
13 last name, the card number used, as well as the MCC code for the suspected fraud.

14 **[0034]** With respect to Fig. 7C, an HEDO user interface 703a may display
15 incremental progress as the HEDO server queries the HEDO database. In one
16 embodiment, the client inputs such as which query to execute, which issuer to query for,
17 and/or the like, e.g., 703b, may be displayed. In other embodiments, differing criterion
18 such as that for a high-end retailer of electronics merchandise may be applied and
19 matching results displayed 703c.

20 **[0035]** With respect to Fig. 7D, the HEDO user interface 704a, may be utilized to
21 perform a consumer loyalty comparison for a merchant against other merchants in a
22 cohort. In one embodiment, competitors of the merchant may be displayed 704b, as

1 well as two queries 704c. In the first query, the merchant may desire to determine their
2 own loyalty score. In one embodiment, in a second query the merchant may desire to
3 determine customer loyalty for its customers that shop at the merchant's competitors.
4 In such cases, the merchant may not have access to the direct underlying transaction
5 amount values for the purchases its customers make at competing merchants. In one
6 embodiment, customer names as well as spend amounts with both the merchant and the
7 merchant competitors may be displayed 704d. In some embodiments, a table may be
8 used to compare, by customer, the merchant spend, market spend, total spend, and/or
9 the like 704e.

10 **[0036]** With respect to Fig. 7E, in some embodiments, the HEDO user interface
11 705a may display a graphical representation that charts consumer loyalty between the
12 merchant and the market as a whole, e.g., 705b. The market in this case being defined
13 as identified competitors of the merchant in question. A relative value of an individual
14 consumer's loyalty to the merchant and to the merchant's competitors may be shown by
15 dividing an aggregate spend representation into multiple sections, e.g. 705c.

16 **[0037]** With respect to Fig. 7F, the HEDO user interface 706a may be utilized by a
17 merchant to compare the merchant's customer risk level to the customer risk level of its
18 competitors. This comparison may be performed by the HEDO server and database
19 without exposing underlying encrypted data to the merchant, and while performing
20 operations on stored encrypted data itself. In one embodiment, competitors of the
21 merchant may be selected 706b and two queries may be created 706c. In the first query
22 the merchant may desire to learn its risk level. In the second query, the risk level
23 associated with the comparison group may be selected and returned. In one

1 embodiment, a merchant may receive a listing of risk levels and the number of its
2 customers at the given risk level 706d. In some embodiments, similar summary data
3 regarding risk levels and customer accounts may be displayed for the comparison
4 merchant group 706e. In some embodiments, a table may be used to chart risk levels of
5 the merchant and the comparison group, e.g., 706f.

6 **[0038]** With respect to Fig. 7G, the HEDO user interface 707a may display a
7 graphical representation comparing risk levels for a merchant and market. Percentile
8 values may be graphed for each risk level for both the merchant and the market, e.g.
9 707b.

10 HEDO Controller

11 **[0039]** Fig. 8 shows a block diagram illustrating embodiments of a HEDO
12 controller. In this embodiment, the HEDO controller 801 may serve to aggregate,
13 process, store, search, serve, identify, instruct, generate, match, and/or facilitate
14 interactions with a computer through various technologies, and/or other related data.

15 **[0040]** Typically, users, which may be people and/or other systems, may engage
16 information technology systems (e.g., computers) to facilitate information processing.
17 In turn, computers employ processors to process information; such processors 803 may
18 be referred to as central processing units (CPU). One form of processor is referred to as
19 a microprocessor. CPUs use communicative circuits to pass binary encoded signals
20 acting as instructions to enable various operations. These instructions may be
21 operational and/or data instructions containing and/or referencing other instructions
22 and data in various processor accessible and operable areas of memory 829 (e.g.,

1 registers, cache memory, random access memory, etc.). Such communicative
2 instructions may be stored and/or transmitted in batches (e.g., batches of instructions)
3 as programs and/or data components to facilitate desired operations. These stored
4 instruction codes, e.g., programs, may engage the CPU circuit components and other
5 motherboard and/or system components to perform desired operations. One type of
6 program is a computer operating system, which, may be executed by CPU on a
7 computer; the operating system enables and facilitates users to access and operate
8 computer information technology and resources. Some resources that may be employed
9 in information technology systems include: input and output mechanisms through
10 which data may pass into and out of a computer; memory storage into which data may
11 be saved; and processors by which information may be processed. These information
12 technology systems may be used to collect data for later retrieval, analysis, and
13 manipulation, which may be facilitated through a database program. These information
14 technology systems provide interfaces that allow users to access and operate various
15 system components.

16 **[0041]** In one embodiment, the HEDO controller 801 may be connected to and/or
17 communicate with entities such as, but not limited to: one or more users from user
18 input devices 811; peripheral devices 812; an optional cryptographic processor device
19 828; and/or a communications network 813.

20 **[0042]** Networks are commonly thought to comprise the interconnection and
21 interoperation of clients, servers, and intermediary nodes in a graph topology. It should
22 be noted that the term “server” as used throughout this application refers generally to a
23 computer, other device, program, or combination thereof that processes and responds to

1 the requests of remote users across a communications network. Servers serve their
2 information to requesting “clients.” The term “client” as used herein refers generally to
3 a computer, program, other device, user and/or combination thereof that is capable of
4 processing and making requests and obtaining and processing any responses from
5 servers across a communications network. A computer, other device, program, or
6 combination thereof that facilitates, processes information and requests, and/or
7 furthers the passage of information from a source user to a destination user is
8 commonly referred to as a “node.” Networks are generally thought to facilitate the
9 transfer of information from source points to destinations. A node specifically tasked
10 with furthering the passage of information from a source to a destination is commonly
11 called a “router.” There are many forms of networks such as Local Area Networks
12 (LANs), Pico networks, Wide Area Networks (WANs), Wireless Networks (WLANs), etc.
13 For example, the Internet is generally accepted as being an interconnection of a
14 multitude of networks whereby remote clients and servers may access and interoperate
15 with one another.

16 **[0043]** The HEDO controller 801 may be based on computer systems that may
17 comprise, but are not limited to, components such as: a computer systemization 802
18 connected to memory 829.

19 **Computer Systemization**

20 **[0044]** A computer systemization 802 may comprise a clock 830, central
21 processing unit (“CPU(s)” and/or “processor(s)” (these terms are used interchangeable
22 throughout the disclosure unless noted to the contrary)) 803, a memory 829 (e.g., a read
23 only memory (ROM) 806, a random access memory (RAM) 805, etc.), and/or an

1 interface bus 807, and most frequently, although not necessarily, are all interconnected
2 and/or communicating through a system bus 804 on one or more (mother)board(s) 802
3 having conductive and/or otherwise transportive circuit pathways through which
4 instructions (e.g., binary encoded signals) may travel to effectuate communications,
5 operations, storage, etc. The computer systemization may be connected to a power
6 source 886; e.g., optionally the power source may be internal. Optionally, a
7 cryptographic processor 826 and/or transceivers (e.g., ICs) 874 may be connected to the
8 system bus. In another embodiment, the cryptographic processor and/or transceivers
9 may be connected as either internal and/or external peripheral devices 812 via the
10 interface bus I/O. In turn, the transceivers may be connected to antenna(s) 875, thereby
11 effectuating wireless transmission and reception of various communication and/or
12 sensor protocols; for example the antenna(s) may connect to: a Texas Instruments
13 WiLink WL1283 transceiver chip (e.g., providing 802.11n, Bluetooth 3.0, FM, global
14 positioning system (GPS) (thereby allowing HEDO controller to determine its
15 location)); Broadcom BCM4329FKUBG transceiver chip (e.g., providing 802.11n,
16 Bluetooth 2.1 + EDR, FM, etc.); a Broadcom BCM4750IUB8 receiver chip (e.g., GPS); an
17 Infineon Technologies X-Gold 618-PMB9800 (e.g., providing 2G/3G HSDPA/HSUPA
18 communications); and/or the like. The system clock typically has a crystal oscillator and
19 generates a base signal through the computer systemization's circuit pathways. The
20 clock is typically coupled to the system bus and various clock multipliers that will
21 increase or decrease the base operating frequency for other components interconnected
22 in the computer systemization. The clock and various components in a computer
23 systemization drive signals embodying information throughout the system. Such
24 transmission and reception of instructions embodying information throughout a

1 computer systemization may be commonly referred to as communications. These
2 communicative instructions may further be transmitted, received, and the cause of
3 return and/or reply communications beyond the instant computer systemization to:
4 communications networks, input devices, other computer systemizations, peripheral
5 devices, and/or the like. It should be understood that in alternative embodiments, any
6 of the above components may be connected directly to one another, connected to the
7 CPU, and/or organized in numerous variations employed as exemplified by various
8 computer systems.

9 **[0045]** The CPU comprises at least one high-speed data processor adequate to
10 execute program components for executing user and/or system-generated requests.
11 Often, the processors themselves will incorporate various specialized processing units,
12 such as, but not limited to: integrated system (bus) controllers, memory management
13 control units, floating point units, and even specialized processing sub-units like
14 graphics processing units, digital signal processing units, and/or the like. Additionally,
15 processors may include internal fast access addressable memory, and be capable of
16 mapping and addressing memory 829 beyond the processor itself; internal memory may
17 include, but is not limited to: fast registers, various levels of cache memory (e.g., level 1,
18 2, 3, etc.), RAM, etc. The processor may access this memory through the use of a
19 memory address space that is accessible via instruction address, which the processor
20 can construct and decode allowing it to access a circuit path to a specific memory
21 address space having a memory state. The CPU may be a microprocessor such as:
22 AMD's Athlon, Duron and/or Opteron; ARM's application, embedded and secure
23 processors; IBM and/or Motorola's DragonBall and PowerPC; IBM's and Sony's Cell

1 processor; Intel's Celeron, Core (2) Duo, Itanium, Pentium, Xeon, and/or XScale;
2 and/or the like processor(s). The CPU interacts with memory through instruction
3 passing through conductive and/or transportive conduits (e.g., (printed) electronic
4 and/or optic circuits) to execute stored instructions (i.e., program code) according to
5 conventional data processing techniques. Such instruction passing facilitates
6 communication within the HEDO controller and beyond through various interfaces.
7 Should processing requirements dictate a greater amount speed and/or capacity,
8 distributed processors (e.g., Distributed HEDO), mainframe, multi-core, parallel,
9 and/or super-computer architectures may similarly be employed. Alternatively, should
10 deployment requirements dictate greater portability, smaller Personal Digital Assistants
11 (PDAs) may be employed.

12 **[0046]** Depending on the particular implementation, features of the HEDO may
13 be achieved by implementing a microcontroller such as CAST's R8051XC2
14 microcontroller; Intel's MCS 51 (i.e., 8051 microcontroller); and/or the like. Also, to
15 implement certain features of the HEDO, some feature implementations may rely on
16 embedded components, such as: Application-Specific Integrated Circuit ("ASIC"),
17 Digital Signal Processing ("DSP"), Field Programmable Gate Array ("FPGA"), and/or the
18 like embedded technology. For example, any of the HEDO component collection
19 (distributed or otherwise) and/or features may be implemented via the microprocessor
20 and/or via embedded components; e.g., via ASIC, coprocessor, DSP, FPGA, and/or the
21 like. Alternately, some implementations of the HEDO may be implemented with
22 embedded components that are configured and used to achieve a variety of features or
23 signal processing.

1 **[0047]** Depending on the particular implementation, the embedded components
2 may include software solutions, hardware solutions, and/or some combination of both
3 hardware/software solutions. For example, HEDO features discussed herein may be
4 achieved through implementing FPGAs, which are a semiconductor devices containing
5 programmable logic components called "logic blocks", and programmable
6 interconnects, such as the high performance FPGA Virtex series and/or the low cost
7 Spartan series manufactured by Xilinx. Logic blocks and interconnects can be
8 programmed by the customer or designer, after the FPGA is manufactured, to
9 implement any of the HEDO features. A hierarchy of programmable interconnects allow
10 logic blocks to be interconnected as needed by the HEDO system
11 designer/administrator, somewhat like a one-chip programmable breadboard. An
12 FPGA's logic blocks can be programmed to perform the operation of basic logic gates
13 such as AND, and XOR, or more complex combinational operators such as decoders or
14 mathematical operations. In most FPGAs, the logic blocks also include memory
15 elements, which may be circuit flip-flops or more complete blocks of memory. In some
16 circumstances, the HEDO may be developed on regular FPGAs and then migrated into a
17 fixed version that more resembles ASIC implementations. Alternate or coordinating
18 implementations may migrate HEDO controller features to a final ASIC instead of or in
19 addition to FPGAs. Depending on the implementation all of the aforementioned
20 embedded components and microprocessors may be considered the "CPU" and/or
21 "processor" for the HEDO.

Power Source

1
2 **[0048]** The power source 886 may be of any standard form for powering small
3 electronic circuit board devices such as the following power cells: alkaline, lithium
4 hydride, lithium ion, lithium polymer, nickel cadmium, solar cells, and/or the like.
5 Other types of AC or DC power sources may be used as well. In the case of solar cells, in
6 one embodiment, the case provides an aperture through which the solar cell may
7 capture photonic energy. The power cell 886 is connected to at least one of the
8 interconnected subsequent components of the HEDO thereby providing an electric
9 current to all subsequent components. In one example, the power source 886 is
10 connected to the system bus component 804. In an alternative embodiment, an outside
11 power source 886 is provided through a connection across the I/O 808 interface. For
12 example, a USB and/or IEEE 1394 connection carries both data and power across the
13 connection and is therefore a suitable source of power.

Interface Adapters

14
15 **[0049]** Interface bus(es) 807 may accept, connect, and/or communicate to a
16 number of interface adapters, conventionally although not necessarily in the form of
17 adapter cards, such as but not limited to: input output interfaces (I/O) 808, storage
18 interfaces 809, network interfaces 810, and/or the like. Optionally, cryptographic
19 processor interfaces 827 similarly may be connected to the interface bus. The interface
20 bus provides for the communications of interface adapters with one another as well as
21 with other components of the computer systemization. Interface adapters are adapted
22 for a compatible interface bus. Interface adapters conventionally connect to the
23 interface bus via a slot architecture. Conventional slot architectures may be employed,

1 such as, but not limited to: Accelerated Graphics Port (AGP), Card Bus, (Extended)
2 Industry Standard Architecture ((E)ISA), Micro Channel Architecture (MCA), NuBus,
3 Peripheral Component Interconnect (Extended) (PCI(X)), PCI Express, Personal
4 Computer Memory Card International Association (PCMCIA), and/or the like.

5 **[0050]** Storage interfaces 809 may accept, communicate, and/or connect to a
6 number of storage devices such as, but not limited to: storage devices 814, removable
7 disc devices, and/or the like. Storage interfaces may employ connection protocols such
8 as, but not limited to: (Ultra) (Serial) Advanced Technology Attachment (Packet
9 Interface) ((Ultra) (Serial) ATA(PI)), (Enhanced) Integrated Drive Electronics ((E)IDE),
10 Institute of Electrical and Electronics Engineers (IEEE) 1394, fiber channel, Small
11 Computer Systems Interface (SCSI), Universal Serial Bus (USB), and/or the like.

12 **[0051]** Network interfaces 810 may accept, communicate, and/or connect to a
13 communications network 813. Through a communications network 813, the HEDO
14 controller is accessible through remote clients 833b (e.g., computers with web browsers)
15 by users 833a. Network interfaces may employ connection protocols such as, but not
16 limited to: direct connect, Ethernet (thick, thin, twisted pair 10/100/1000 Base T,
17 and/or the like), Token Ring, wireless connection such as IEEE 802.11a-x, and/or the
18 like. Should processing requirements dictate a greater amount speed and/or capacity,
19 distributed network controllers (e.g., Distributed HEDO), architectures may similarly be
20 employed to pool, load balance, and/or otherwise increase the communicative
21 bandwidth required by the HEDO controller. A communications network may be any
22 one and/or the combination of the following: a direct interconnection; the Internet; a
23 Local Area Network (LAN); a Metropolitan Area Network (MAN); an Operating

1 Missions as Nodes on the Internet (OMNI); a secured custom connection; a Wide Area
2 Network (WAN); a wireless network (e.g., employing protocols such as, but not limited
3 to a Wireless Application Protocol (WAP), I-mode, and/or the like); and/or the like. A
4 network interface may be regarded as a specialized form of an input output interface.
5 Further, multiple network interfaces 810 may be used to engage with various
6 communications network types 813. For example, multiple network interfaces may be
7 employed to allow for the communication over broadcast, multicast, and/or unicast
8 networks.

9 **[0052]** Input Output interfaces (I/O) 808 may accept, communicate, and/or
10 connect to user input devices 811, peripheral devices 812, cryptographic processor
11 devices 828, and/or the like. I/O may employ connection protocols such as, but not
12 limited to: audio: analog, digital, monaural, RCA, stereo, and/or the like; data: Apple
13 Desktop Bus (ADB), IEEE 1394a-b, serial, universal serial bus (USB); infrared; joystick;
14 keyboard; midi; optical; PC AT; PS/2; parallel; radio; video interface: Apple Desktop
15 Connector (ADC), BNC, coaxial, component, composite, digital, Digital Visual Interface
16 (DVI), high-definition multimedia interface (HDMI), RCA, RF antennae, S-Video, VGA,
17 and/or the like; wireless transceivers: 802.11a/b/g/n/x; Bluetooth; cellular (e.g., code
18 division multiple access (CDMA), high speed packet access (HSPA(+)), high-speed
19 downlink packet access (HSDPA), global system for mobile communications (GSM),
20 long term evolution (LTE), WiMax, etc.); and/or the like. One typical output device may
21 include a video display, which typically comprises a Cathode Ray Tube (CRT) or Liquid
22 Crystal Display (LCD) based monitor with an interface (e.g., DVI circuitry and cable)
23 that accepts signals from a video interface, may be used. The video interface composites

1 information generated by a computer systemization and generates video signals based
2 on the composited information in a video memory frame. Another output device is a
3 television set, which accepts signals from a video interface. Typically, the video interface
4 provides the composited video information through a video connection interface that
5 accepts a video display interface (e.g., an RCA composite video connector accepting an
6 RCA composite video cable; a DVI connector accepting a DVI display cable, etc.).

7 **[0053]** User input devices 811 often are a type of peripheral device 512 (see below)
8 and may include: card readers, dongles, finger print readers, gloves, graphics tablets,
9 joysticks, keyboards, microphones, mouse (mice), remote controls, retina readers, touch
10 screens (e.g., capacitive, resistive, etc.), trackballs, trackpads, sensors (e.g.,
11 accelerometers, ambient light, GPS, gyroscopes, proximity, etc.), styluses, and/or the
12 like.

13 **[0054]** Peripheral devices 812 may be connected and/or communicate to I/O
14 and/or other facilities of the like such as network interfaces, storage interfaces, directly
15 to the interface bus, system bus, the CPU, and/or the like. Peripheral devices may be
16 external, internal and/or part of the HEDO controller. Peripheral devices may include:
17 antenna, audio devices (e.g., line-in, line-out, microphone input, speakers, etc.),
18 cameras (e.g., still, video, webcam, etc.), dongles (e.g., for copy protection, ensuring
19 secure transactions with a digital signature, and/or the like), external processors (for
20 added capabilities; e.g., crypto devices 528), force-feedback devices (e.g., vibrating
21 motors), network interfaces, printers, scanners, storage devices, transceivers (e.g.,
22 cellular, GPS, etc.), video devices (e.g., goggles, monitors, etc.), video sources, visors,
23 and/or the like. Peripheral devices often include types of input devices (e.g., cameras).

1 **[0055]** It should be noted that although user input devices and peripheral devices
2 may be employed, the HEDO controller may be embodied as an embedded, dedicated,
3 and/or monitor-less (i.e., headless) device, wherein access would be provided over a
4 network interface connection.

5 **[0056]** Cryptographic units such as, but not limited to, microcontrollers,
6 processors 826, interfaces 827, and/or devices 828 may be attached, and/or
7 communicate with the HEDO controller. A MC68HC16 microcontroller, manufactured
8 by Motorola Inc., may be used for and/or within cryptographic units. The MC68HC16
9 microcontroller utilizes a 16-bit multiply-and-accumulate instruction in the 16 MHz
10 configuration and requires less than one second to perform a 512-bit RSA private key
11 operation. Cryptographic units support the authentication of communications from
12 interacting agents, as well as allowing for anonymous transactions. Cryptographic units
13 may also be configured as part of the CPU. Equivalent microcontrollers and/or
14 processors may also be used. Other commercially available specialized cryptographic
15 processors include: Broadcom's CryptoNetX and other Security Processors; nCipher's
16 nShield; SafeNet's Luna PCI (e.g., 7100) series; Semaphore Communications' 40 MHz
17 Roadrunner 184; Sun's Cryptographic Accelerators (e.g., Accelerator 6000 PCIe Board,
18 Accelerator 500 Daughtercard); Via Nano Processor (e.g., L2100, L2200, U2400) line,
19 which is capable of performing 500+ MB/s of cryptographic instructions; VLSI
20 Technology's 33 MHz 6868; and/or the like.

21

Memory

22 **[0057]** Generally, any mechanization and/or embodiment allowing a processor to
23 affect the storage and/or retrieval of information is regarded as memory 829. However,

1 memory is a fungible technology and resource, thus, any number of memory
2 embodiments may be employed in lieu of or in concert with one another. It is to be
3 understood that the HEDO controller and/or a computer systemization may employ
4 various forms of memory 829. For example, a computer systemization may be
5 configured wherein the operation of on-chip CPU memory (e.g., registers), RAM, ROM,
6 and any other storage devices are provided by a paper punch tape or paper punch card
7 mechanism; however, such an embodiment would result in an extremely slow rate of
8 operation. In a typical configuration, memory 829 will include ROM 806, RAM 805,
9 and a storage device 814. A storage device 814 may be any conventional computer
10 system storage. Storage devices may include a drum; a (fixed and/or removable)
11 magnetic disk drive; a magneto-optical drive; an optical drive (i.e., Blu-ray, CD
12 ROM/RAM/Recordable (R)/ReWritable (RW), DVD R/RW, HD DVD R/RW etc.); an
13 array of devices (e.g., Redundant Array of Independent Disks (RAID)); solid state
14 memory devices (USB memory, solid state drives (SSD), etc.); other processor-readable
15 storage mediums; and/or other devices of the like. Thus, a computer systemization
16 generally requires and makes use of memory.

17 **Component Collection**

18 **[0058]** The memory 829 may contain a collection of program and/or database
19 components and/or data such as, but not limited to: operating system component(s) 815
20 (operating system); information server component(s) 816 (information server); user
21 interface component(s) 817 (user interface); Web browser component(s) 818 (Web
22 browser); database(s) 819; mail server component(s) 821; mail client component(s)
23 822; cryptographic server component(s) 820 (cryptographic server); the HEDO

1 component(s) 835; HJ Component 841; HOP Component 842; GHP Component 843;
2 and/or the like (i.e., collectively a component collection). These components may be
3 stored and accessed from the storage devices and/or from storage devices accessible
4 through an interface bus. Although non-conventional program components such as
5 those in the component collection, typically, are stored in a local storage device 814, they
6 may also be loaded and/or stored in memory such as: peripheral devices, RAM, remote
7 storage facilities through a communications network, ROM, various forms of memory,
8 and/or the like.

9 **Operating System**

10 **[0059]** The operating system component 815 is an executable program
11 component facilitating the operation of the HEDO controller. Typically, the operating
12 system facilitates access of I/O, network interfaces, peripheral devices, storage devices,
13 and/or the like. The operating system may be a highly fault tolerant, scalable, and
14 secure system such as: Apple Macintosh OS X (Server); AT&T Plan 9; Be OS; Unix and
15 Unix-like system distributions (such as AT&T's UNIX; Berkley Software Distribution
16 (BSD) variations such as FreeBSD, NetBSD, OpenBSD, and/or the like; Linux
17 distributions such as Red Hat, Ubuntu, and/or the like); and/or the like operating
18 systems. However, more limited and/or less secure operating systems also may be
19 employed such as Apple Macintosh OS, IBM OS/2, Microsoft DOS, Microsoft Windows
20 2000/2003/3.1/95/98/CE/Millennium/NT/Vista/XP/Win7 (Server), Palm OS, and/or
21 the like. An operating system may communicate to and/or with other components in a
22 component collection, including itself, and/or the like. Most frequently, the operating
23 system communicates with other program components, user interfaces, and/or the like.

1 For example, the operating system may contain, communicate, generate, obtain, and/or
2 provide program component, system, user, and/or data communications, requests,
3 and/or responses. The operating system, once executed by the CPU, may enable the
4 interaction with communications networks, data, I/O, peripheral devices, program
5 components, memory, user input devices, and/or the like. The operating system may
6 provide communications protocols that allow the HEDO controller to communicate with
7 other entities through a communications network 813. Various communication
8 protocols may be used by the HEDO controller as a subcarrier transport mechanism for
9 interaction, such as, but not limited to: multicast, TCP/IP, UDP, unicast, and/or the
10 like.

11 Information Server

12 **[0060]** An information server component 816 is a stored program component that
13 is executed by a CPU. The information server may be a conventional Internet
14 information server such as, but not limited to Apache Software Foundation's Apache,
15 Microsoft's Internet Information Server, and/or the like. The information server may
16 allow for the execution of program components through facilities such as Active Server
17 Page (ASP), ActiveX, (ANSI) (Objective-) C (++), C# and/or .NET, Common Gateway
18 Interface (CGI) scripts, dynamic (D) hypertext markup language (HTML), FLASH, Java,
19 JavaScript, Practical Extraction Report Language (PERL), Hypertext Pre-Processor
20 (PHP), pipes, Python, wireless application protocol (WAP), WebObjects, and/or the like.
21 The information server may support secure communications protocols such as, but not
22 limited to, File Transfer Protocol (FTP); HyperText Transfer Protocol (HTTP); Secure
23 Hypertext Transfer Protocol (HTTPS), Secure Socket Layer (SSL), messaging protocols

1 (e.g., America Online (AOL) Instant Messenger (AIM), Application Exchange (APEX),
2 ICQ, Internet Relay Chat (IRC), Microsoft Network (MSN) Messenger Service, Presence
3 and Instant Messaging Protocol (PRIM), Internet Engineering Task Force's (IETF's)
4 Session Initiation Protocol (SIP), SIP for Instant Messaging and Presence Leveraging
5 Extensions (SIMPLE), open XML-based Extensible Messaging and Presence Protocol
6 (XMPP) (i.e., Jabber or Open Mobile Alliance's (OMA's) Instant Messaging and
7 Presence Service (IMPS)), Yahoo! Instant Messenger Service, and/or the like. The
8 information server provides results in the form of Web pages to Web browsers, and
9 allows for the manipulated generation of the Web pages through interaction with other
10 program components. After a Domain Name System (DNS) resolution portion of an
11 HTTP request is resolved to a particular information server, the information server
12 resolves requests for information at specified locations on the HEDO controller based
13 on the remainder of the HTTP request. For example, a request such as
14 `http://123.124.125.126/myInformation.html` might have the IP portion of the request
15 "123.124.125.126" resolved by a DNS server to an information server at that IP address;
16 that information server might in turn further parse the http request for the
17 `"/myInformation.html"` portion of the request and resolve it to a location in memory
18 containing the information "myInformation.html." Additionally, other information
19 serving protocols may be employed across various ports, e.g., FTP communications
20 across port 21, and/or the like. An information server may communicate to and/or with
21 other components in a component collection, including itself, and/or facilities of the
22 like. Most frequently, the information server communicates with the HEDO database
23 819, operating systems, other program components, user interfaces, Web browsers,
24 and/or the like.

1 **[0061]** Access to the HEDO database may be achieved through a number of
2 database bridge mechanisms such as through scripting languages as enumerated below
3 (e.g., CGI) and through inter-application communication channels as enumerated below
4 (e.g., CORBA, WebObjects, etc.). Any data requests through a Web browser are parsed
5 through the bridge mechanism into appropriate grammars as required by the HEDO. In
6 one embodiment, the information server would provide a Web form accessible by a Web
7 browser. Entries made into supplied fields in the Web form are tagged as having been
8 entered into the particular fields, and parsed as such. The entered terms are then passed
9 along with the field tags, which act to instruct the parser to generate queries directed to
10 appropriate tables and/or fields. In one embodiment, the parser may generate queries in
11 standard SQL by instantiating a search string with the proper join/select commands
12 based on the tagged text entries, wherein the resulting command is provided over the
13 bridge mechanism to the HEDO as a query. Upon generating query results from the
14 query, the results are passed over the bridge mechanism, and may be parsed for
15 formatting and generation of a new results Web page by the bridge mechanism. Such a
16 new results Web page is then provided to the information server, which may supply it to
17 the requesting Web browser.

18 **[0062]** Also, an information server may contain, communicate, generate, obtain,
19 and/or provide program component, system, user, and/or data communications,
20 requests, and/or responses.

21

User Interface

22 **[0063]** Computer interfaces in some respects are similar to automobile operation
23 interfaces. Automobile operation interface elements such as steering wheels, gearshifts,

1 and speedometers facilitate the access, operation, and display of automobile resources,
2 and status. Computer interaction interface elements such as check boxes, cursors,
3 menus, scrollers, and windows (collectively and commonly referred to as widgets)
4 similarly facilitate the access, capabilities, operation, and display of data and computer
5 hardware and operating system resources, and status. Operation interfaces are
6 commonly called user interfaces. Graphical user interfaces (GUIs) such as the Apple
7 Macintosh Operating System's Aqua, IBM's OS/2, Microsoft's Windows
8 2000/2003/3.1/95/98/CE/Millennium/NT/XP/Vista/7 (i.e., Aero), Unix's X-Windows
9 (e.g., which may include additional Unix graphic interface libraries and layers such as K
10 Desktop Environment (KDE), mythTV and GNU Network Object Model Environment
11 (GNOME)), web interface libraries (e.g., ActiveX, AJAX, (D)HTML, FLASH, Java,
12 JavaScript, etc. interface libraries such as, but not limited to, Dojo, jQuery UI,
13 MooTools, Prototype, script.aculo.us, SWFObject, Yahoo! User Interface, any of which
14 may be used and provide a baseline and means of accessing and displaying information
15 graphically to users.

16 **[0064]** A user interface component 817 is a stored program component that is
17 executed by a CPU. The user interface may be a conventional graphic user interface as
18 provided by, with, and/or atop operating systems and/or operating environments such
19 as already discussed. The user interface may allow for the display, execution,
20 interaction, manipulation, and/or operation of program components and/or system
21 facilities through textual and/or graphical facilities. The user interface provides a facility
22 through which users may affect, interact, and/or operate a computer system. A user
23 interface may communicate to and/or with other components in a component

1 collection, including itself, and/or facilities of the like. Most frequently, the user
2 interface communicates with operating systems, other program components, and/or the
3 like. The user interface may contain, communicate, generate, obtain, and/or provide
4 program component, system, user, and/or data communications, requests, and/or
5 responses.

6 **Web Browser**

7 **[0065]** A Web browser component 818 is a stored program component that is
8 executed by a CPU. The Web browser may be a conventional hypertext viewing
9 application such as Microsoft Internet Explorer or Netscape Navigator. Secure Web
10 browsing may be supplied with 128bit (or greater) encryption by way of HTTPS, SSL,
11 and/or the like. Web browsers allowing for the execution of program components
12 through facilities such as ActiveX, AJAX, (D)HTML, FLASH, Java, JavaScript, web
13 browser plug-in APIs (e.g., Firefox, Safari Plug-in, and/or the like APIs), and/or the like.
14 Web browsers and like information access tools may be integrated into PDAs, cellular
15 telephones, and/or other mobile devices. A Web browser may communicate to and/or
16 with other components in a component collection, including itself, and/or facilities of
17 the like. Most frequently, the Web browser communicates with information servers,
18 operating systems, integrated program components (e.g., plug-ins), and/or the like; e.g.,
19 it may contain, communicate, generate, obtain, and/or provide program component,
20 system, user, and/or data communications, requests, and/or responses. Also, in place of
21 a Web browser and information server, a combined application may be developed to
22 perform similar operations of both. The combined application would similarly affect the
23 obtaining and the provision of information to users, user agents, and/or the like from

1 the HEDO enabled nodes. The combined application may be nugatory on systems
2 employing standard Web browsers.

3 **Mail Server**

4 **[0066]** A mail server component 821 is a stored program component that is
5 executed by a CPU 803. The mail server may be a conventional Internet mail server such
6 as, but not limited to sendmail, Microsoft Exchange, and/or the like. The mail server
7 may allow for the execution of program components through facilities such as ASP,
8 ActiveX, (ANSI) (Objective-) C (++), C# and/or .NET, CGI scripts, Java, JavaScript,
9 PERL, PHP, pipes, Python, WebObjects, and/or the like. The mail server may support
10 communications protocols such as, but not limited to: Internet message access protocol
11 (IMAP), Messaging Application Programming Interface (MAPI)/Microsoft Exchange,
12 post office protocol (POP3), simple mail transfer protocol (SMTP), and/or the like. The
13 mail server can route, forward, and process incoming and outgoing mail messages that
14 have been sent, relayed and/or otherwise traversing through and/or to the HEDO.

15 **[0067]** Access to the HEDO mail may be achieved through a number of APIs
16 offered by the individual Web server components and/or the operating system.

17 **[0068]** Also, a mail server may contain, communicate, generate, obtain, and/or
18 provide program component, system, user, and/or data communications, requests,
19 information, and/or responses.

20 **Mail Client**

21 **[0069]** A mail client component 822 is a stored program component that is
22 executed by a CPU 803. The mail client may be a conventional mail viewing application

1 such as Apple Mail, Microsoft Entourage, Microsoft Outlook, Microsoft Outlook
2 Express, Mozilla, Thunderbird, and/or the like. Mail clients may support a number of
3 transfer protocols, such as: IMAP, Microsoft Exchange, POP3, SMTP, and/or the like. A
4 mail client may communicate to and/or with other components in a component
5 collection, including itself, and/or facilities of the like. Most frequently, the mail client
6 communicates with mail servers, operating systems, other mail clients, and/or the like;
7 e.g., it may contain, communicate, generate, obtain, and/or provide program
8 component, system, user, and/or data communications, requests, information, and/or
9 responses. Generally, the mail client provides a facility to compose and transmit
10 electronic mail messages.

11 **Cryptographic Server**

12 **[0070]** A cryptographic server component 820 is a stored program component
13 that is executed by a CPU 803, cryptographic processor 826, cryptographic processor
14 interface 827, cryptographic processor device 828, and/or the like. Cryptographic
15 processor interfaces will allow for expedition of encryption and/or decryption requests
16 by the cryptographic component; however, the cryptographic component, alternatively,
17 may run on a conventional CPU. The cryptographic component allows for the
18 encryption and/or decryption of provided data. The cryptographic component allows for
19 both symmetric and asymmetric (e.g., Pretty Good Protection (PGP)) encryption and/or
20 decryption. The cryptographic component may employ cryptographic techniques such
21 as, but not limited to: digital certificates (e.g., X.509 authentication framework), digital
22 signatures, dual signatures, enveloping, password access protection, public key
23 management, and/or the like. The cryptographic component will facilitate numerous

1 (encryption and/or decryption) security protocols such as, but not limited to: checksum,
2 Data Encryption Standard (DES), Elliptical Curve Encryption (ECC), International Data
3 Encryption Algorithm (IDEA), Message Digest 5 (MD5, which is a one way hash
4 operation), passwords, Rivest Cipher (RC5), Rijndael, RSA (which is an Internet
5 encryption and authentication system that uses an algorithm developed in 1977 by Ron
6 Rivest, Adi Shamir, and Leonard Adleman), Secure Hash Algorithm (SHA), Secure
7 Socket Layer (SSL), Secure Hypertext Transfer Protocol (HTTPS), and/or the like.
8 Employing such encryption security protocols, the HEDO may encrypt all incoming
9 and/or outgoing communications and may serve as node within a virtual private
10 network (VPN) with a wider communications network. The cryptographic component
11 facilitates the process of “security authorization” whereby access to a resource is
12 inhibited by a security protocol wherein the cryptographic component effects authorized
13 access to the secured resource. In addition, the cryptographic component may provide
14 unique identifiers of content, e.g., employing and MD5 hash to obtain a unique
15 signature for an digital audio file. A cryptographic component may communicate to
16 and/or with other components in a component collection, including itself, and/or
17 facilities of the like. The cryptographic component supports encryption schemes
18 allowing for the secure transmission of information across a communications network
19 to enable the HEDO component to engage in secure transactions if so desired. The
20 cryptographic component facilitates the secure accessing of resources on the HEDO and
21 facilitates the access of secured resources on remote systems; i.e., it may act as a client
22 and/or server of secured resources. Most frequently, the cryptographic component
23 communicates with information servers, operating systems, other program components,
24 and/or the like. The cryptographic component may contain, communicate, generate,

1 obtain, and/or provide program component, system, user, and/or data communications,
2 requests, and/or responses.

3 **The HEDO Database**

4 **[0071]** The HEDO database component 819 may be embodied in a database and
5 its stored data. The database is a stored program component, which is executed by the
6 CPU; the stored program component portion configuring the CPU to process the stored
7 data. The database may be a conventional, fault tolerant, relational, scalable, secure
8 database such as Oracle or Sybase. Relational databases are an extension of a flat file.
9 Relational databases consist of a series of related tables. The tables are interconnected
10 via a key field. Use of the key field allows the combination of the tables by indexing
11 against the key field; i.e., the key fields act as dimensional pivot points for combining
12 information from various tables. Relationships generally identify links maintained
13 between tables by matching primary keys. Primary keys represent fields that uniquely
14 identify the rows of a table in a relational database. More precisely, they uniquely
15 identify rows of a table on the “one” side of a one-to-many relationship.

16 **[0072]** Alternatively, the HEDO database may be implemented using various
17 standard data-structures, such as an array, hash, (linked) list, struct, structured text file
18 (e.g., XML), table, and/or the like. Such data-structures may be stored in memory
19 and/or in (structured) files. In another alternative, an object-oriented database may be
20 used, such as Frontier, ObjectStore, Poet, Zope, and/or the like. Object databases can
21 include a number of object collections that are grouped and/or linked together by
22 common attributes; they may be related to other object collections by some common
23 attributes. Object-oriented databases perform similarly to relational databases with the

1 exception that objects are not just pieces of data but may have other types of capabilities
2 encapsulated within a given object. If the HEDO database is implemented as a data-
3 structure, the use of the HEDO database 819 may be integrated into another component
4 such as the HEDO component 835. Also, the database may be implemented as a mix of
5 data structures, objects, and relational structures. Databases may be consolidated
6 and/or distributed in countless variations through standard data processing techniques.
7 Portions of databases, e.g., tables, may be exported and/or imported and thus
8 decentralized and/or integrated.

9 **[0073]** In one embodiment, the database component 819 includes several tables
10 819a-m. A Users table 819a may include fields such as, but not limited to: user_id, ssn,
11 dob, first_name, last_name, age, state, address_firstline, address_secondline, zipcode,
12 devices_list, contact_info, contact_type, alt_contact_info, alt_contact_type, and/or the
13 like. The Users table may support and/or track multiple entity accounts on a HEDO. A
14 Clients table 819b may include fields such as, but not limited to: client_id, client_name,
15 client_ip, client_type, client_model, operating_system, os_version, app_installed_flag,
16 and/or the like. An Apps table 819c may include fields such as, but not limited to:
17 app_id, app_name, app_type, os_compatibilities_list, version, timestamp,
18 developer_id, and/or the like. A Merchants table 819d may include fields such as, but
19 not limited to: merchant_id, merchant_name, merchant_address, ip_address,
20 mac_address, auth_key, port_num, security_settings_list, and/or the like. An Issuers
21 table 819e may include fields such as, but not limited to: issuer_id, issuer_name,
22 issuer_address, ip_address, mac_address, auth_key, port_num, security_settings_list,
23 and/or the like. An Acquirers table 819f may include fields such as, but not limited to:

1 acquirer_id, acquirer_name, acquirer_gateway_id, issuer_acquirer_flag,
2 institution_name, and/or the like. An Accounts table 819g may include fields such as,
3 but not limited to: account_id, account_firstname, account_lastname, account_type,
4 account_num, account_balance_list, billingaddress_line1, billingaddress_line2,
5 billing_zipcode, billing_state, shipping_preferences, shippingaddress_line1,
6 shippingaddress_line2, shipping_zipcode, shipping_state, and/or the like. A
7 Transactions table 819h may include fields such as, but not limited to: transaction_id,
8 user_id, timestamp, transaction_cost, purchase_details_list, num_products,
9 products_list, product_type, product_params_list, product_title, product_summary,
10 quantity, account_firstname, account_lastname, account_type, account_num,
11 billingaddress_line1, billingaddress_line2, billing_zipcode, billing_state,
12 shipping_preferences, shippingaddress_line1, shippingaddress_line2,
13 shipping_zipcode, shipping_state, merchant_id, merchant_name, merchant_auth_
14 key, and/or the like. An Encryption table 819i may include fields such as, but not
15 limited to: encryption_id, encryption_scheme, encryption_method, salt_value,
16 key_ids, and/or the like. A Models table 819j may include fields such as, but not limited
17 to: model_id, model_name, user_id, issuer_id, merchant_id, model_description,
18 model_commands, and/or the like. A Keys table 819k may include fields such as, but
19 not limited to: key_id, key_decrypts_record_pointer, generated_date, owned_by_id,
20 user_id, acquirer_id, merchant_id, server_identifier, key_expiration, and/or the like.
21 A Grouping Keys table 819l may include fields such as, but not limited to:
22 grouping_key_id, sub_key_of_key_id, key_id, description, server_identifier, table_id,
23 table_names, table_grouping_fields, table_grouping_rangees, and/or the like. A Table
24 Schemas table 819m may include fields such as, but not limited to: table_schema_id,

1 server_identifier, table_names, table_structure, grouping_method,
2 table_encrypted_values, table_encrypted_value_methods, and/or the like.

3 **[0074]** In one embodiment, the HEDO database may interact with other database
4 systems. For example, employing a distributed database system, queries and data access
5 by search HEDO component may treat the combination of the HEDO database, an
6 integrated data security layer database as a single database entity.

7 **[0075]** In one embodiment, user programs may contain various user interface
8 primitives, which may serve to update the HEDO. Also, various accounts may require
9 custom database tables depending upon the environments and the types of clients the
10 HEDO may need to serve. It should be noted that any unique fields may be designated
11 as a key field throughout. In an alternative embodiment, these tables have been
12 decentralized into their own databases and their respective database controllers (i.e.,
13 individual database controllers for each of the above tables). Employing standard data
14 processing techniques, one may further distribute the databases over several computer
15 systemizations and/or storage devices. Similarly, configurations of the decentralized
16 database controllers may be varied by consolidating and/or distributing the various
17 database components 819a-m. The HEDO may be configured to keep track of various
18 settings, inputs, and parameters via database controllers.

19 **[0076]** The HEDO database may communicate to and/or with other components
20 in a component collection, including itself, and/or facilities of the like. Most frequently,
21 the HEDO database communicates with the HEDO component, other program

1 components, and/or the like. The database may contain, retain, and provide
2 information regarding other nodes and data.

The HEDOs

4 **[0077]** The HEDO component 835 is a stored program component that is
5 executed by a CPU. In one embodiment, the HEDO component incorporates any and/or
6 all combinations of the aspects of the HEDO that was discussed in the previous figures.
7 As such, the HEDO affects accessing, obtaining and the provision of information,
8 services, transactions, and/or the like across various communications networks. The
9 features and embodiments of the HEDO discussed herein increase network efficiency by
10 reducing data transfer requirements the use of more efficient data structures and
11 mechanisms for their transfer and storage. As a consequence, more data may be
12 transferred in less time, and latencies with regard to transactions, are also reduced. In
13 many cases, such reduction in storage, transfer time, bandwidth requirements,
14 latencies, etc., will reduce the capacity and structural infrastructure requirements to
15 support the HEDO's features and facilities, and in many cases reduce the costs, energy
16 consumption/requirements, and extend the life of HEDO's underlying infrastructure;
17 this has the added benefit of making the HEDO more reliable. Similarly, many of the
18 features and mechanisms are designed to be easier for users to use and access, thereby
19 broadening the audience that may enjoy/employ and exploit the feature sets of the
20 HEDO; such ease of use also helps to increase the reliability of the HEDO. In addition,
21 the feature sets include heightened security as noted via the Cryptographic components
22 820, 826, 828 and throughout, making access to the features and data more reliable and
23 secure.

1 **[0078]** The HEDO component may transform transaction storage requests and
2 model query requests, and/or the like and use the HEDO. In one embodiment, the
3 HEDO component 835 takes inputs (e.g., transaction storage request 203a, 203b,
4 HEDO package storage request 208, homomorphic insertion 212, model query request
5 303, request grouping keys 306, HEDO model query request 309, HEDO query 312
6 and/or the like) etc., and transforms the inputs via various components (e.g., HJ
7 Component 841; HOP Component 842; GHP Component 843, and/or the like), into
8 outputs (e.g., HEDO results response 317, HEDO model query results 319, receive
9 grouping keys 308, model query response 321, render model query output 322, and/or
10 the like).

11 **[0079]** The HEDO component enabling access of information between nodes may
12 be developed by employing standard development tools and languages such as, but not
13 limited to: Apache components, Assembly, ActiveX, binary executables, (ANSI)
14 (Objective-) C (++), C# and/or .NET, database adapters, CGI scripts, Java, JavaScript,
15 mapping tools, procedural and object oriented development tools, PERL, PHP, Python,
16 shell scripts, SQL commands, web application server extensions, web development
17 environments and libraries (e.g., Microsoft's ActiveX; Adobe AIR, FLEX & FLASH;
18 AJAX; (D)HTML; Dojo, Java; JavaScript; jQuery(UI); MooTools; Prototype;
19 script.aculo.us; Simple Object Access Protocol (SOAP); SWFObject; Yahoo! User
20 Interface; and/or the like), WebObjects, and/or the like. In one embodiment, the HEDO
21 server employs a cryptographic server to encrypt and decrypt communications. The
22 HEDO component may communicate to and/or with other components in a component
23 collection, including itself, and/or facilities of the like. Most frequently, the HEDO

1 component communicates with the HEDO database, operating systems, other program
2 components, and/or the like. The HEDO may contain, communicate, generate, obtain,
3 and/or provide program component, system, user, and/or data communications,
4 requests, and/or responses.

5 **Distributed HEDOs**

6 **[0080]** The structure and/or operation of any of the HEDO node controller
7 components may be combined, consolidated, and/or distributed in any number of ways
8 to facilitate development and/or deployment. Similarly, the component collection may
9 be combined in any number of ways to facilitate deployment and/or development. To
10 accomplish this, one may integrate the components into a common code base or in a
11 facility that can dynamically load the components on demand in an integrated fashion.

12 **[0081]** The component collection may be consolidated and/or distributed in
13 countless variations through standard data processing and/or development techniques.
14 Multiple instances of any one of the program components in the program component
15 collection may be instantiated on a single node, and/or across numerous nodes to
16 improve performance through load-balancing and/or data-processing techniques.
17 Furthermore, single instances may also be distributed across multiple controllers
18 and/or storage devices; e.g., databases. All program component instances and
19 controllers working in concert may do so through standard data processing
20 communication techniques.

21 **[0082]** The configuration of the HEDO controller will depend on the context of
22 system deployment. Factors such as, but not limited to, the budget, capacity, location,

1 and/or use of the underlying hardware resources may affect deployment requirements
2 and configuration. Regardless of if the configuration results in more consolidated
3 and/or integrated program components, results in a more distributed series of program
4 components, and/or results in some combination between a consolidated and
5 distributed configuration, data may be communicated, obtained, and/or provided.
6 Instances of components consolidated into a common code base from the program
7 component collection may communicate, obtain, and/or provide data. This may be
8 accomplished through intra-application data processing communication techniques
9 such as, but not limited to: data referencing (e.g., pointers), internal messaging, object
10 instance variable communication, shared memory space, variable passing, and/or the
11 like.

12 **[0083]** If component collection components are discrete, separate, and/or
13 external to one another, then communicating, obtaining, and/or providing data with
14 and/or to other component components may be accomplished through inter-application
15 data processing communication techniques such as, but not limited to: Application
16 Program Interfaces (API) information passage; (distributed) Component Object Model
17 ((D)COM), (Distributed) Object Linking and Embedding ((D)OLE), and/or the like),
18 Common Object Request Broker Architecture (CORBA), Jini local and remote
19 application program interfaces, JavaScript Object Notation (JSON), Remote Method
20 Invocation (RMI), SOAP, process pipes, shared files, and/or the like. Messages sent
21 between discrete component components for inter-application communication or within
22 memory spaces of a singular component for intra-application communication may be
23 facilitated through the creation and parsing of a grammar. A grammar may be

1 developed by using development tools such as lex, yacc, XML, and/or the like, which
2 allow for grammar generation and parsing capabilities, which in turn may form the basis
3 of communication messages within and between components.

4 **[0084]** For example, a grammar may be arranged to recognize the tokens of an
5 HTTP post command, e.g.:

6 `w3c -post http://... Value1`

7 **[0085]** where Value1 is discerned as being a parameter because “http://” is part of
8 the grammar syntax, and what follows is considered part of the post value. Similarly,
9 with such a grammar, a variable “Value1” may be inserted into an “http://” post
10 command and then sent. The grammar syntax itself may be presented as structured data
11 that is interpreted and/or otherwise used to generate the parsing mechanism (e.g., a
12 syntax description text file as processed by lex, yacc, etc.). Also, once the parsing
13 mechanism is generated and/or instantiated, it itself may process and/or parse
14 structured data such as, but not limited to: character (e.g., tab) delineated text, HTML,
15 structured text streams, XML, and/or the like structured data. In another embodiment,
16 inter-application data processing protocols themselves may have integrated and/or
17 readily available parsers (e.g., JSON, SOAP, and/or like parsers) that may be employed
18 to parse (e.g., communications) data. Further, the parsing grammar may be used
19 beyond message parsing, but may also be used to parse: databases, data collections, data
20 stores, structured data, and/or the like. Again, the desired configuration will depend
21 upon the context, environment, and requirements of system deployment.

1 **[0086]** For example, in some implementations, the HEDO controller may be
2 executing a PHP script implementing a Secure Sockets Layer (“SSL”) socket server via
3 the information sherver, which listens to incoming communications on a server port to
4 which a client may send data, e.g., data encoded in JSON format. Upon identifying an
5 incoming communication, the PHP script may read the incoming message from the
6 client device, parse the received JSON-encoded text data to extract information from the
7 JSON-encoded text data into PHP script variables, and store the data (e.g., client
8 identifying information, etc.) and/or extracted information in a relational database
9 accessible using the Structured Query Language (“SQL”). An exemplary listing, written
10 substantially in the form of PHP/SQL commands, to accept JSON-encoded input data
11 from a client device via a SSL connection, parse the data to extract variables, and store
12 the data to a database, is provided below:

```
13  <?PHP
14  header('Content-Type: text/plain');
15
16  //set ip address and port to listen to for incoming data
17  $address = '192.168.0.100';
18  $port = 255;
19
20  //create a server-side SSL socket, listen
21  //for/accept incoming communication
22  $sock = socket_create(AF_INET, SOCK_STREAM, 0);
23  socket_bind($sock, $address, $port)
24      or die('Could not bind to address');
25  socket_listen($sock);
26  $client = socket_accept($sock);
27
28  //read input data from client device in 1024 byte
29  //blocks until end of message
```

```
1  do {
2      $input = "";
3      $input = socket_read($client, 1024);
4      $data .= $input;
5  } while($input != "");
6
7  // parse data to extract variables
8  $obj = json_decode($data, true);
9
10 // store input data in a database
11 mysql_connect("10.1.1.1",$srvr,$pass); // access database server
12 mysql_select("CLIENT_DB.SQL"); // select database to append
13 mysql_query("INSERT INTO UserTable (transmission) VALUES
14 ($data)"); // add data to UserTable table in a CLIENT database
15 mysql_close("CLIENT_DB.SQL"); // close connection to database
16 ?>
```

17 **[0087]** Also, the following resources may be used to provide example
18 embodiments regarding SOAP parser implementation:

```
19 http://www.xav.com/perl/site/lib/SOAP/Parser.html
20 http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?
21 topic=/com.ibm.IBMDI.doc/referenceguide295.htm
```

22 **[0088]** and other parser implementations:

```
23 http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?
24 topic=/com.ibm.IBMDI.doc/referenceguide259.htm
```

25
26 **[0089]** all of which are hereby expressly incorporated by reference.

27 **[0090]** In order to address various issues and advance the art, the entirety of this
28 application for HEDO (including the Cover Page, Title, Headings, Field, Background,
29 Summary, Brief Description of the Drawings, Detailed Description, Claims, Abstract,

1 Figures, Appendices, and otherwise) shows, by way of illustration, various embodiments
2 in which the claimed innovations may be practiced. The advantages and features of the
3 application are of a representative sample of embodiments only, and are not exhaustive
4 and/or exclusive. They are presented only to assist in understanding and teach the
5 claimed principles. It should be understood that they are not representative of all
6 claimed innovations. As such, certain aspects of the disclosure have not been discussed
7 herein. That alternate embodiments may not have been presented for a specific portion
8 of the innovations or that further undescribed alternate embodiments may be available
9 for a portion is not to be considered a disclaimer of those alternate embodiments. It will
10 be appreciated that many of those undescribed embodiments incorporate the same
11 principles of the innovations and others are equivalent. Thus, it is to be understood that
12 other embodiments may be utilized and functional, logical, operational, organizational,
13 structural and/or topological modifications may be made without departing from the
14 scope and/or spirit of the disclosure. As such, all examples and/or embodiments are
15 deemed to be non-limiting throughout this disclosure. Also, no inference should be
16 drawn regarding those embodiments discussed herein relative to those not discussed
17 herein other than it is as such for purposes of reducing space and repetition. For
18 instance, it is to be understood that the logical and/or topological structure of any
19 combination of any program components (a component collection), other components
20 and/or any present feature sets as described in the figures and/or throughout are not
21 limited to a fixed operating order and/or arrangement, but rather, any disclosed order is
22 exemplary and all equivalents, regardless of order, are contemplated by the disclosure.
23 Furthermore, it is to be understood that such features are not limited to serial execution,
24 but rather, any number of threads, processes, services, servers, and/or the like that may

1 execute asynchronously, concurrently, in parallel, simultaneously, synchronously,
2 and/or the like are contemplated by the disclosure. As such, some of these features may
3 be mutually contradictory, in that they cannot be simultaneously present in a single
4 embodiment. Similarly, some features are applicable to one aspect of the innovations,
5 and inapplicable to others. In addition, the disclosure includes other innovations not
6 presently claimed. Applicant reserves all rights in those presently unclaimed
7 innovations including the right to claim such innovations, file additional applications,
8 continuations, continuations in part, divisions, and/or the like thereof. As such, it
9 should be understood that advantages, embodiments, examples, functional, features,
10 logical, operational, organizational, structural, topological, and/or other aspects of the
11 disclosure are not to be considered limitations on the disclosure as defined by the claims
12 or limitations on equivalents to the claims. It is to be understood that, depending on the
13 particular needs and/or characteristics of a HEDO individual and/or enterprise user,
14 database configuration and/or relational model, data type, data transmission and/or
15 network framework, syntax structure, and/or the like, various embodiments of the
16 HEDO, may be implemented that enable a great deal of flexibility and customization.
17 For example, aspects of the HEDO may be adapted for restaurant dining, online
18 shopping ,brick-and-mortar shopping, secured information processing, and/or the like.
19 While various embodiments and discussions of the HEDO have been directed to
20 electronic purchase transactions, however, it is to be understood that the embodiments
21 described herein may be readily configured and/or customized for a wide variety of
22 other applications and/or implementations.

1 **[0091]** As additional examples of the wide scope of the systems and methods
2 disclosed herein, the following example illustrates a fraud scenario. In this example,
3 multiple purchases were made by the same cardholder at the same merchant type
4 (MCC) with cards for multiple issuers. Merchant types were either gasoline/service
5 stations or high-end electronics. Ticket value was \$1500 plus for high-end electronics
6 and \$200 plus for gasoline. Fig. 9 shows an example of targeting fraud using HEDO
7 operations. At step 902, the issuing bank is identified by data collected by Visa. The
8 system retrieves all Visa cards (PANs) issued by the issuing bank at step 904. At step
9 906, the cardholder is identified for each PAN, and all Visa cards held by each
10 cardholder is identified at step 908. Step 910 examines transactions made by all Visa
11 cards held by each cardholder to find fraudulent activity. For suspected fraudulent
12 transactions, step 912 reports issuer surname, PAN, and MCC for further investigation.
13 At step 914, Visa sends similar reports to all other impacted issuers. Fig. 10 depicts a
14 sample representation involving a Visa credit card 1002 and a Visa debit card 1004 that
15 are being handled by merchants. An alert symbol is provided for any identified issues.

16 **[0092]** It should be understood that different or additional embodiments can be
17 considered in such examples. For example, in one embodiment, the systems and
18 methods may use live data including timestamps to add another dimension of analysis.
19 As another example, multi-party agreements may enable account-level analysis, rather
20 than only at the PAN or surname level.

21 **[0093]** Other embodiments include multiple public/private keys being required
22 and/or multiple encryption schemes being used. Software/hardware optimizations can
23 also be used with fully homomorphic encryption. In another embodiment, multi-party

1 relationships can be used to harness the benefits of aggregate data. Still further, the
2 systems and methods disclosed herein can run on modeled data, not live data. In
3 another embodiment, the model strives to encapsulate reality and contains key domain
4 knowledge. Additionally, if fully homomorphic encryption takes an excessively long
5 time to compute, then somewhat homomorphic encryption can be used as described in
6 Appendix A of the following application: United States provisional patent application
7 serial no. 61/861,368, filed August 1, 2013, attorney docket no. 532US01, entitled
8 “Homomorphic Database Operations Apparatuses, Methods and Systems,” which is
9 expressly incorporated herein by reference.

10 **[0094]** As another example, HEDO operations can also be used with customer
11 acquisition. In one scenario, the merchant may be looking for consumers who match its
12 customer demographics by querying transactions made by related merchants. As an
13 example, Fig. 11 illustrates at 1102 analysis involving “Tiffany” offering a new sterling
14 silver tag that it wishes to market to frequent purchasers of coach and/or Louis Vuitton
15 handbags. The analysis using HEDO operations identifies such customers at region
16 1104. This can optimize the merchant’s marketing dollars by developing precise
17 marketing campaigns to reach new customers who are in the merchant’s exact target
18 market. Product-level data can also be incorporated in the analysis. Because of such
19 operations, Visa can provide the conduit for the offer without revealing private customer
20 information. Also, targeted customers could opt-in to the Visa offers program.

21 **[0095]** Other embodiments involving HEDO operations can include (Dis-)Loyalty
22 scenarios. As an illustration, a merchant can compare its customers’ transactions with
23 those made at its competitors. Analysis involving HEDO operations can assess which

1 customers are the most valuable to the target with marketing campaigns to convert to
2 higher transactions versus loyal customers to the company or to its competitors.
3 Reports based upon such analysis can be generated including the graphical report and
4 comparison shown in Fig. 12 at 1202. In one embodiment, information may be made
5 more useful with product-level data rather than with only ticket values. Additionally,
6 implementations using live data could include timestamps and geo-location to further
7 assist in precise targeting.

1 What is claimed is:

2
3 1. An encrypted table value homomorphically joining apparatus, comprising:

4 a memory; and

5 a processor disposed in communication with said memory, and configured
6 to issue a plurality of processing instructions stored in the memory, wherein the
7 processor issues instructions to:

8 receive a model query for processing, at a data repository;

9 determine that the model query requires data values from more
10 than one table in the repository;

11 evaluate the model query component tokens to determine a join
12 type to perform, a plurality of tables to be joined, and a plurality of fields on which to
13 join the plurality of tables;

14 determine that the plurality of fields on which to join includes at
15 least one field that contains homomorphically encrypted data;

16 perform a homomorphic join using the plurality of fields; and
17 provide the resultant homomorphically joined tables.

18
19 2. The apparatus of claim 1, additionally comprising instructions to:

20 determine a homomorphic join strategy.

21
22 3. The apparatus of claim 2, wherein the homomorphic join strategy is directly
23 comparing values in two homomorphically encrypted fields when it is determined that
24 the fields have both been encrypted using a common deterministic encryption scheme.

25
26 4. The apparatus of claim 2, wherein the homomorphic join strategy is using a
27 key common to all values in at least one of the plurality of fields to decrypt the plurality
28 of homomorphically encrypted values in a field.

29
30 5. The apparatus of claim 2, wherein when the homomorphic join strategy is to
31 utilize grouping keys, additionally comprising instructions to:

32 receive a plurality of grouping keys;

1 determine a range of fields values that each grouping key may be used to
2 decrypt; and

3 decrypt each of the values in each of the ranges using the grouping keys.

4
5 6. The apparatus of claim 1, additionally comprising instructions to:
6 aggregate results from the resultant homomorphically joined tables.

7
8 7. The apparatus of claim 6, wherein aggregating involves leaving encrypted
9 space and re-entering space.

10
11 8. The apparatus of claim 6, wherein the aggregation is performed completely in
12 encrypted space.

13
14 9. A processor implemented method of homomorphically joining repository
15 tables using encrypted table values, comprising:

16 receiving a model query for processing, at a data repository;

17 determining that the model query requires data values from more than
18 one table in the repository;

19 evaluating the model query component tokens to determine a join type to
20 perform, a plurality of tables to be joined, and a plurality of fields on which to join the
21 plurality of tables;

22 determining that the plurality of fields on which to join includes at least
23 one field that contains homomorphically encrypted data;

24 performing a homomorphic join using the plurality of fields; and

25 providing the resultant homomorphically joined tables.

26
27 10. The method of claim 9, additionally comprising:

28 determining a homomorphic join strategy.

29
30 11. The method of claim 10, wherein the homomorphic join strategy is directly
31 comparing values in two homomorphically encrypted fields when it is determined that
32 the fields have both been encrypted using a common deterministic encryption scheme.

1
2 12. The method of claim 10, wherein the homomorphic join strategy is using a
3 key common to all values in at least one of the plurality of fields to decrypt the plurality
4 of homomorphically encrypted values in a field.

5
6 13. The method of claim 10, wherein when the homomorphic join strategy is to
7 utilize grouping keys, additionally comprising:

8 receiving a plurality of grouping keys;

9 determining a range of fields values that each grouping key may be used to
10 decrypt; and

11 decrypting each of the values in each of the ranges using the grouping
12 keys.

13
14 14. The method of claim 9, additionally comprising:

15 aggregating results from the resultant homomorphically joined tables.

16
17 15. The method of claim 14, wherein aggregating involves leaving encrypted
18 space and re-entering space.

19
20 16. The method of claim 14, wherein the aggregation is performed completely in
21 encrypted space.

22

23

24

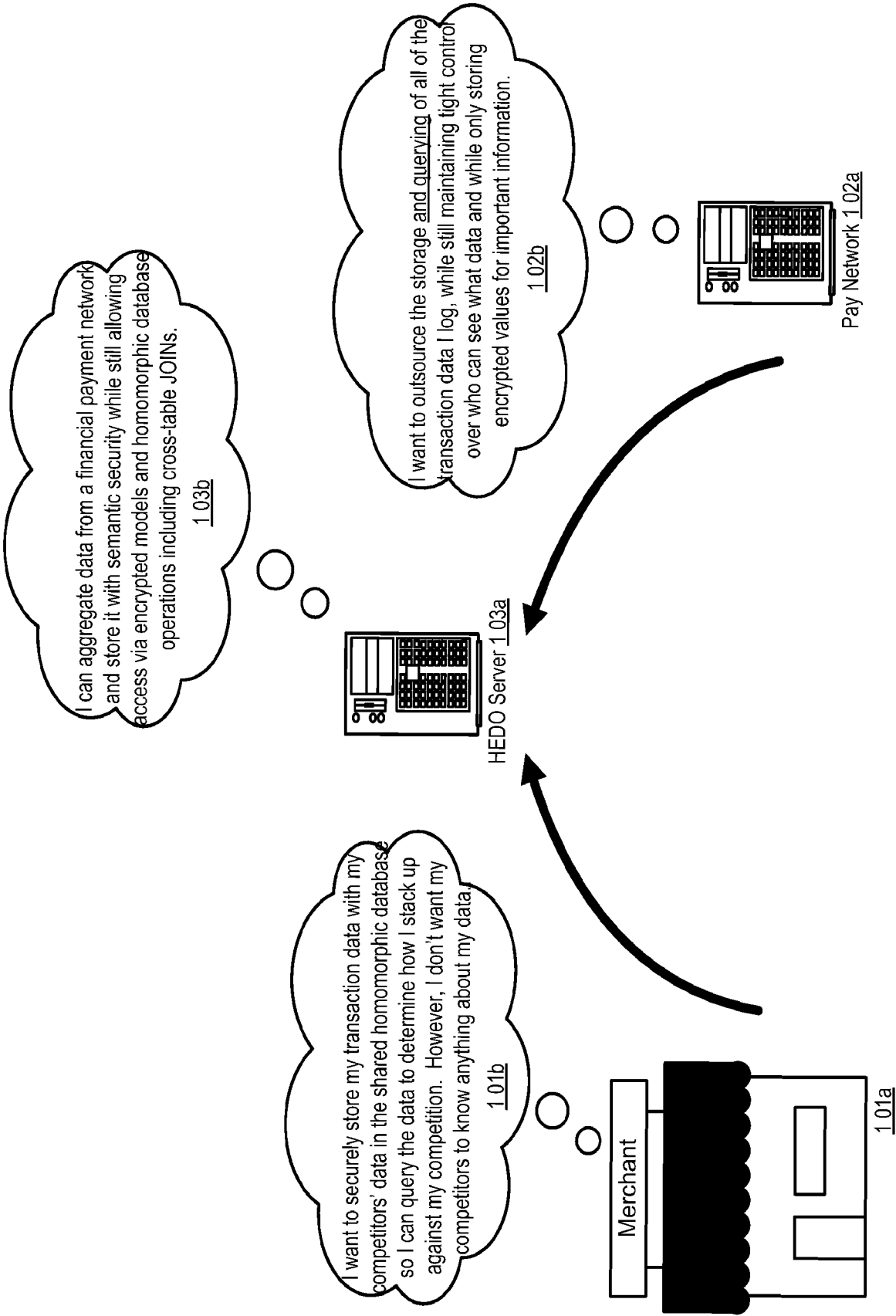


Figure 1

Example: Homomorphically Encrypted Database Operations

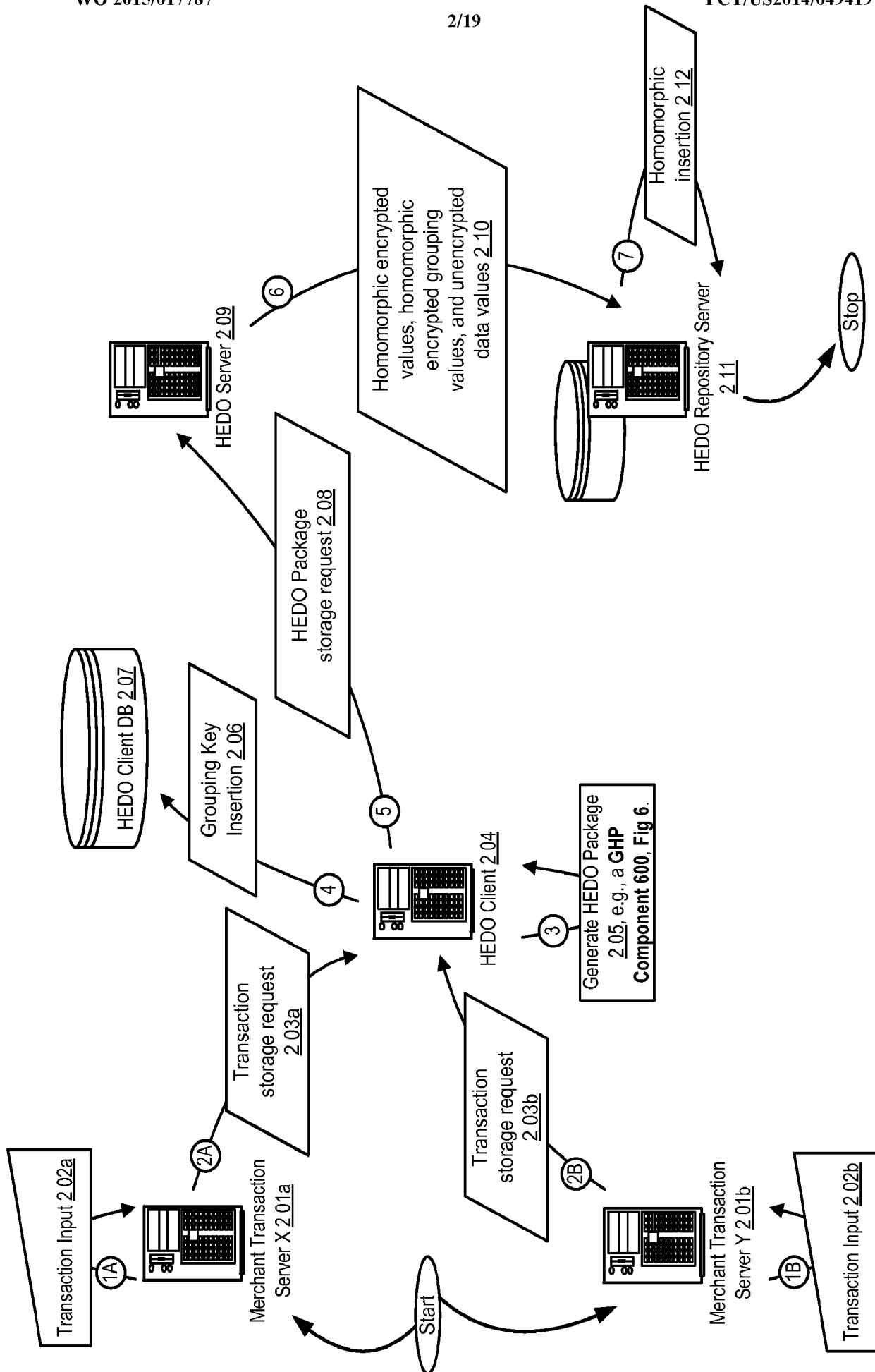


Figure 2

Example Data Flow: Homomorphically Encrypted Database Transaction Data Aggregation

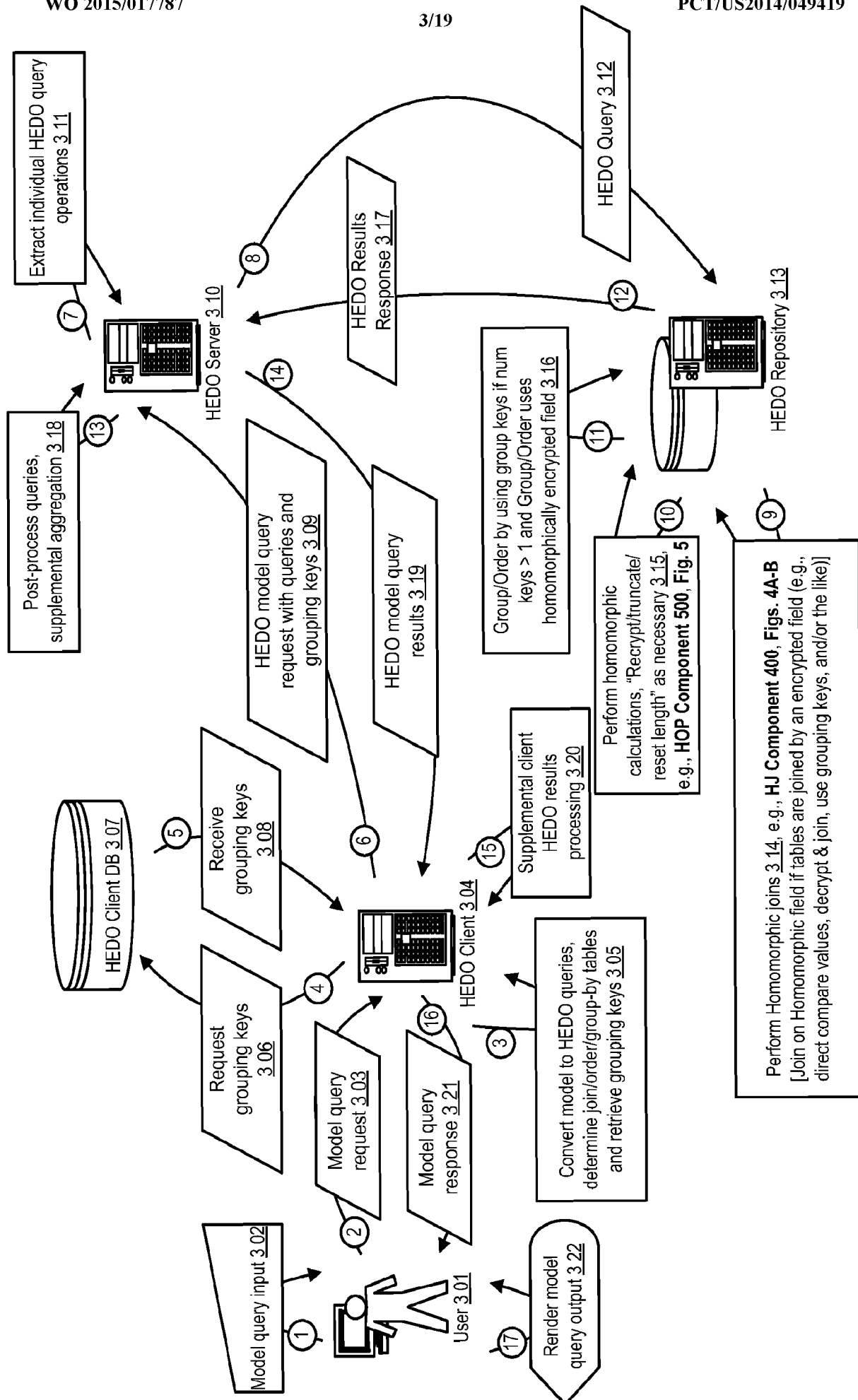
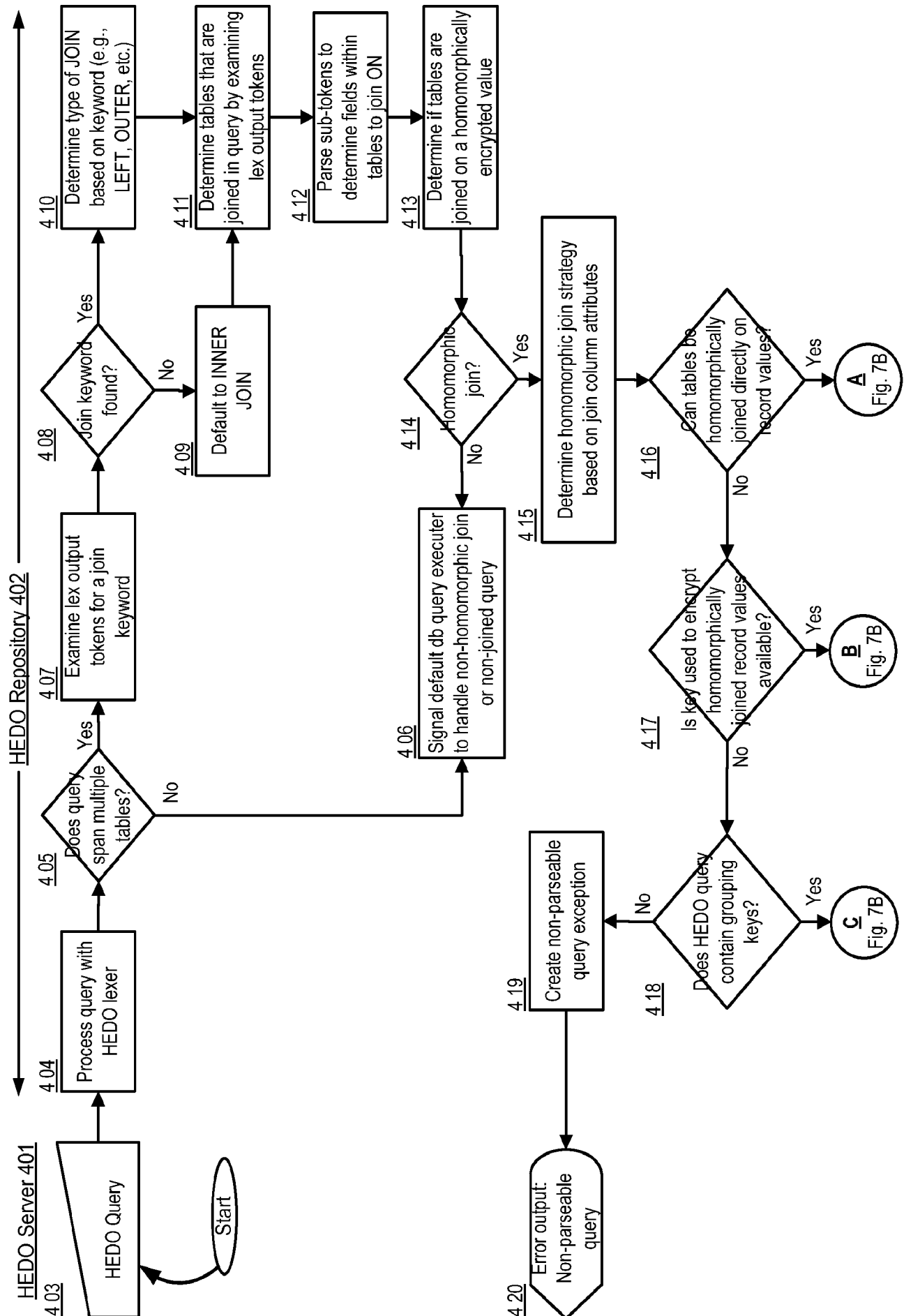


Figure 3

Example Data Flow: Querying and Joining Homomorphically Encrypted Databases



Example Logic Flow: Performing Homomorphic Joins, e.g., HJ Component 400

Figure 4A

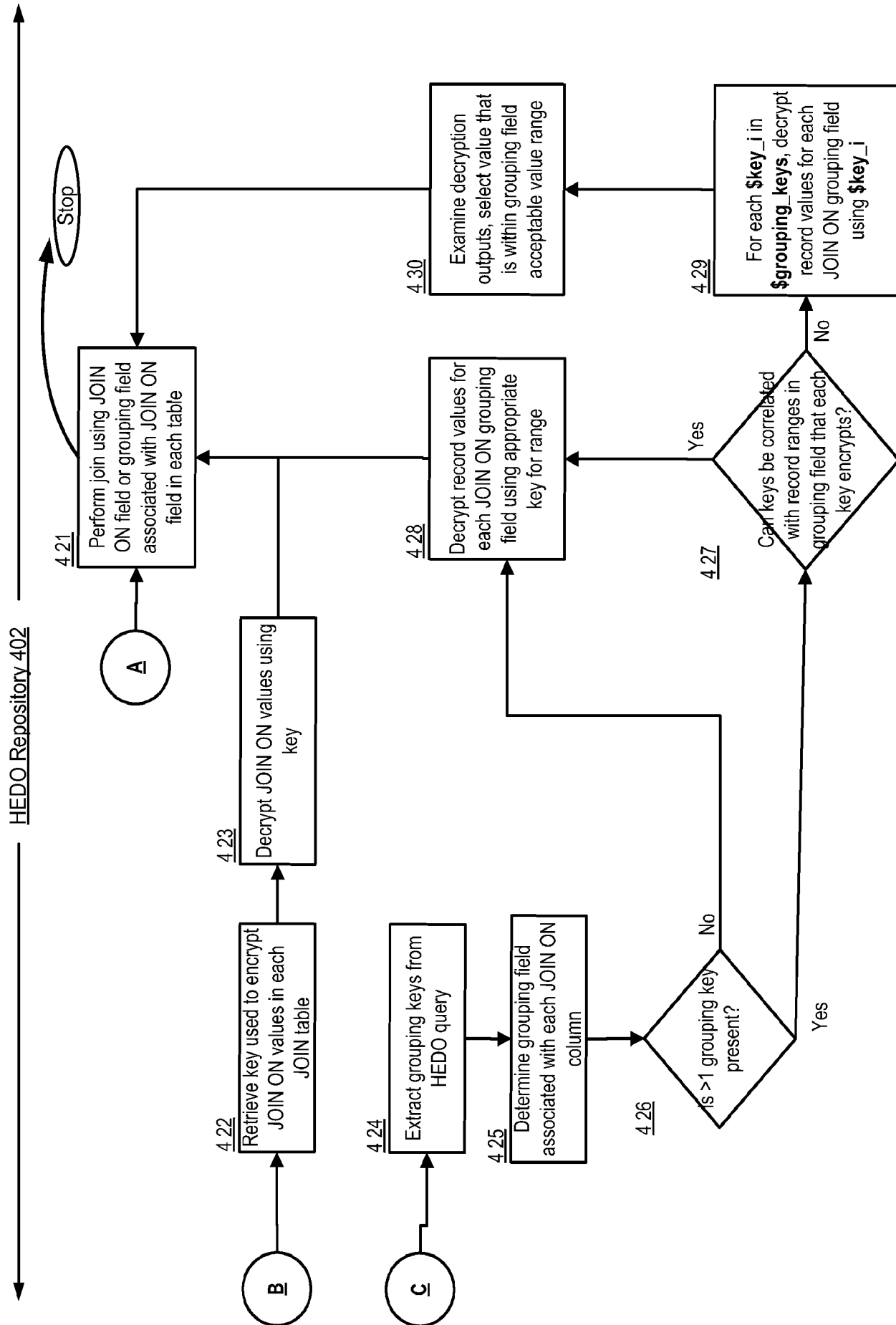


Figure 4B

Example Logic Flow: Performing Homomorphic Joins, e.g., HJ Component 400

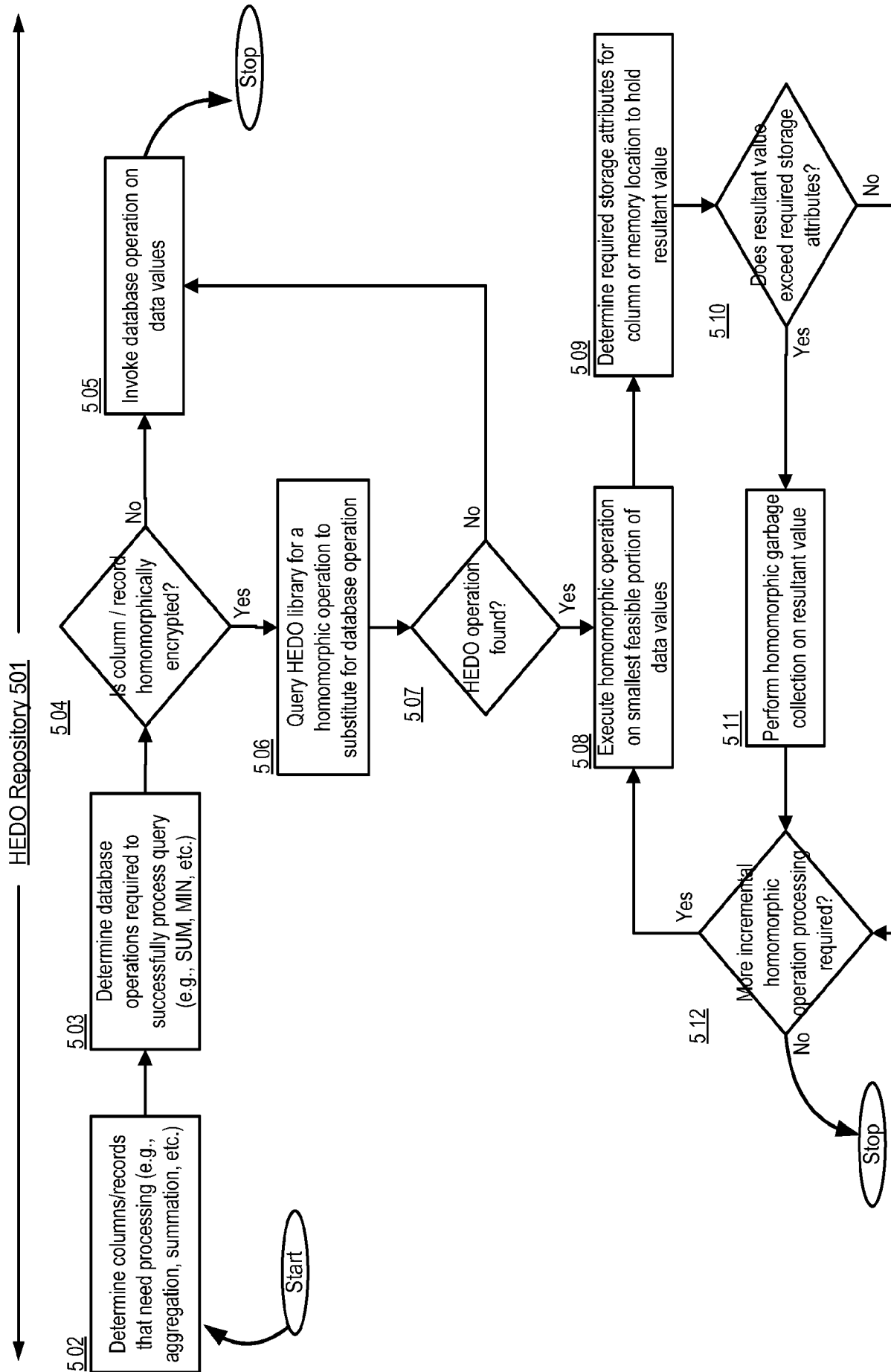


Figure 5

Example Logic Flow: Homomorphic Calculations in Database Space, e.g., HOP Component 500

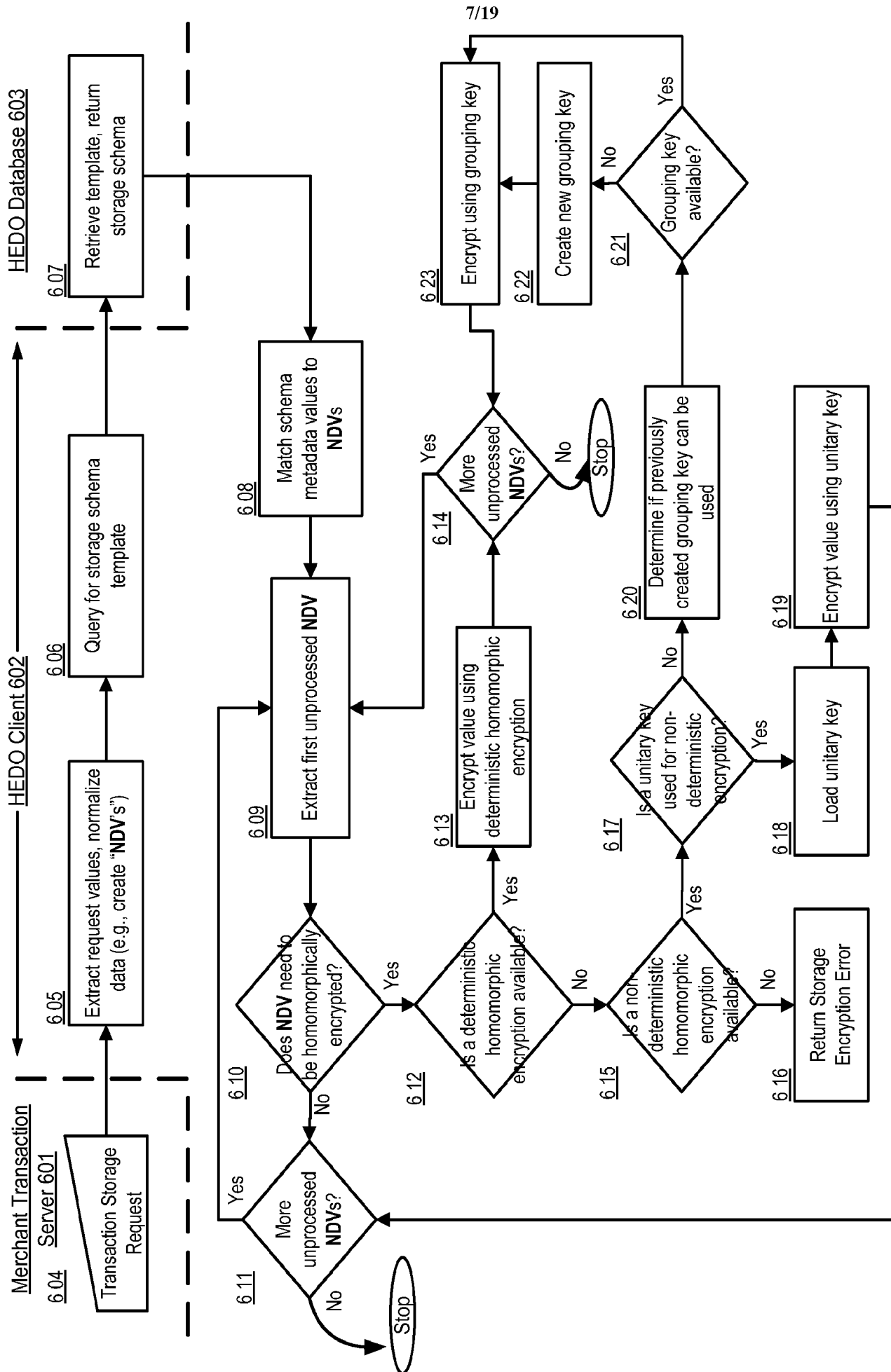


Figure 6

Example Logic Flow: Generating HEDO Packages, e.g. GHP Component 600

7.01a

VISA **FHE DEMO**

1. Fraud Detection
Multiple purchases made by the same cardholder at the same merchant type with cards from multiple issuers.

7.01b

2. Loyalty Comparison
Merchant compares its customers' transactions to those made at its competitors.

7.01c

3. Risk Distribution
Merchant compares its customers' Risk levels to those of its competitors.

7.01d

GO!

A	B	C	D	E

GRAPHS EXIT

Figure 7A

7 02a

VISA FHE DEMO

1. Fraud Detection
Multiple purchases made by the same cardholder at the same merchant type with cards from multiple issuers.

2. Loyalty Comparison
Merchant compares its customers' transactions to those made at its competitors.

3. Risk Distribution
Merchant compares its customers' Risk levels to those of its competitors.

Commonwith WestPac ANZ
NationalBnk CUA

7 02b

```

select distinct CardholderE.Surname,
CardholderE.Number
into MyCustomers from CardholderE
where CardholderE.Issuer = 'Issuer-0000001';
select MyCustomers.Surname, TransactionE.Cardholder,
MerchantE.Category
into Relevant
from TransactionE, MerchantE, MyCustomers
where TransactionE.Merchant = MerchantE.Number
  
```

7 02c

GO!

Customer ACOSTA used card 4000014000005655 at MCC 5651
2 transactions at the same MCC using different cards.
Customer ALEXANDER used card 4000014000002991 at MCC 7349
2 transactions at the same MCC using different cards.
Customer BRADLEY used card 4000014000006190 at MCC 1799
2 transactions at the same MCC using different cards.
Customer BURKE used card 4000014000006711 at MCC 3776
2 transactions at the same MCC using different cards.
Customer CAMPBELL used card 4000014000006323 at MCC 5960

7 02d

CUSTOMER	CARD NO.	MCC CODE
ACOSTA	4000014000005655	5651
ALEXANDER	4000014000002991	7349
BRADLEY	4000014000006190	1799
BURKE	4000014000006711	3776
CAMPBELL	4000014000006323	5960

7 02e

GRAPHS EXIT

Figure 7B

Example: HEDO User Interface

7.03a

Python Shell				X	
File Edit Shell Debug Options Windows Help				-	
Python 2.7.3 (default, 32 Type "copyright", "credits" or "license()" for more information >>> ===== RESTART ===== >>> Standard homomorphically encrypted length of one bit is 3150 bits. This is roughly 949 decimal digits. Client sent: {'Procedure': 'QueryEight', 'Issuer': 'Issuer-000001'} 7.03b 33 potential cases of fraud. Case 1 : total amount in question on 400001000005655 is: 2660.4 Applying criterion for high-end electronics merchants. ** FLAG AS FRAUD ** 7.03c Case 2 : total amount in question on 4000014000002991 is: 2548.7 Applying criterion for high-end electronics merchants. ** FLAG AS FRAUD ** Case 3 : total amount in question on 4000014000006190 is: 2971.5 Applying criterion for high-end electronics merchants. ** FLAG AS FRAUD ** Case 4: total amount in question on 4000014000006711 is: 242.8 Applying criterion for gasoline/service satatations. ** FLAG AS FRAUD ** Case 5 : total amount in question on 4000014000006323 is: 269.28				^	
				v	
				Ln 122 Col 46	

Figure 7C

Example: HEDO User Interface

704a

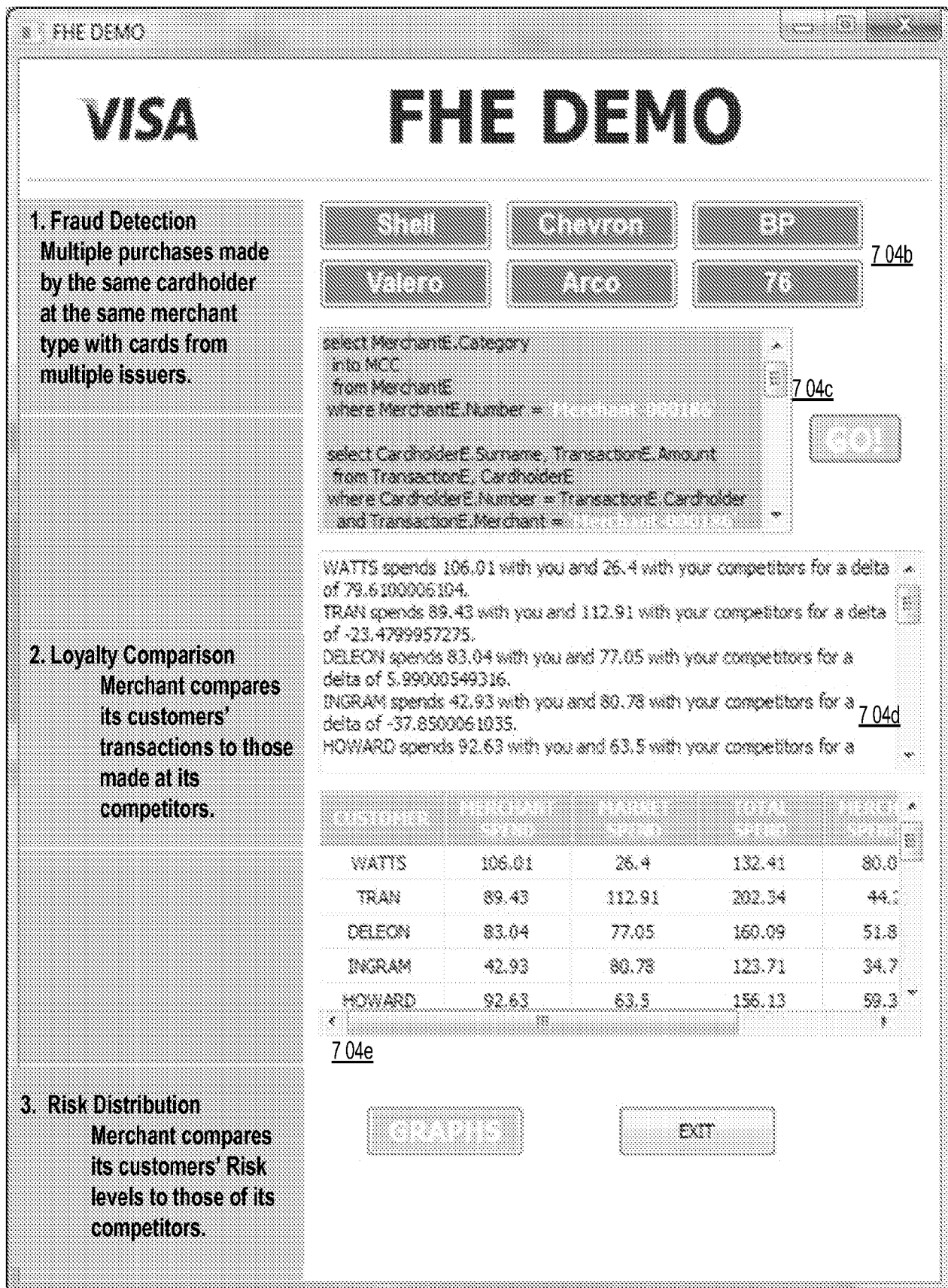


Figure 7D

Example: HEDO User Interface

7 05a

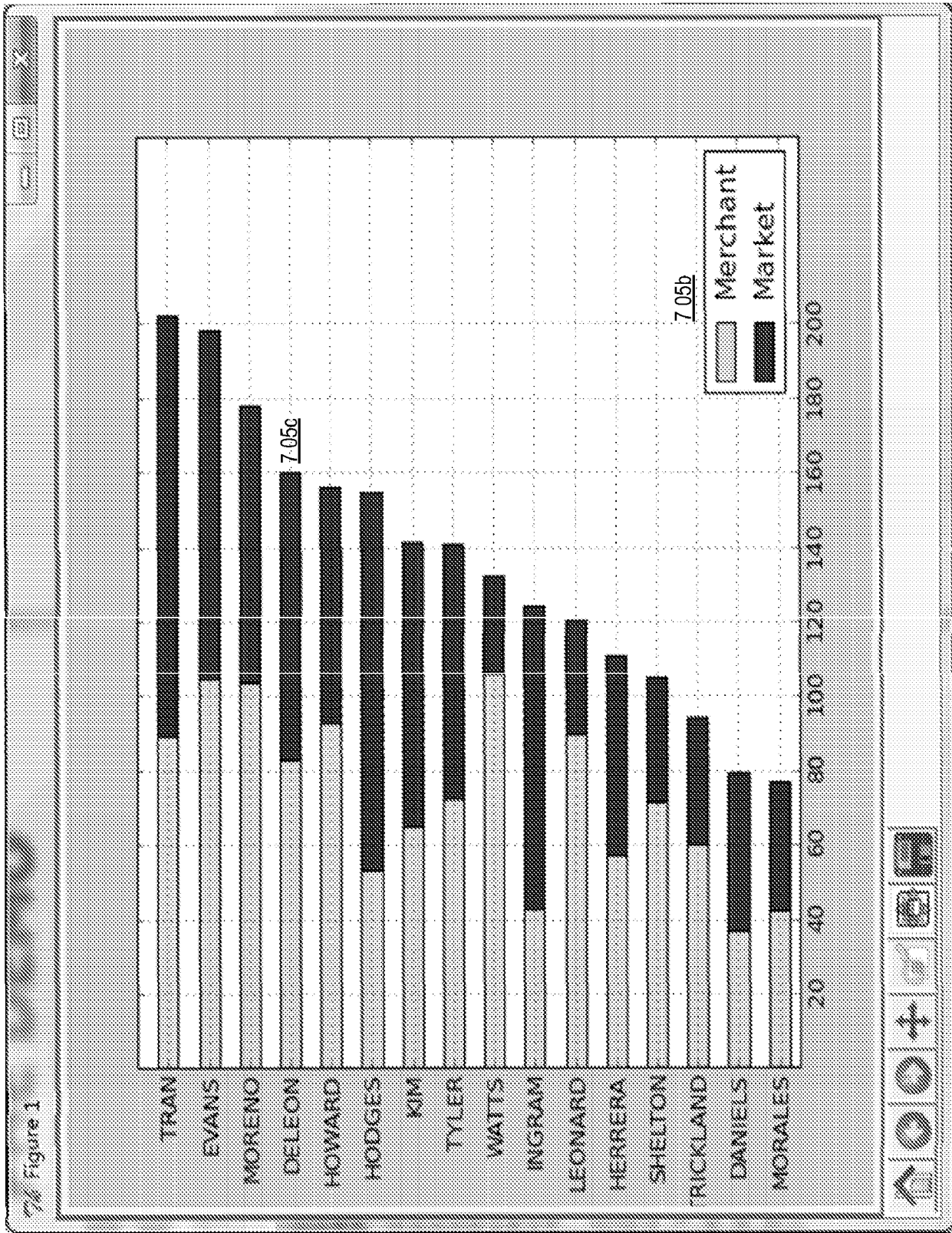


Figure 7E

Example: HEDO User Interface

7.06a

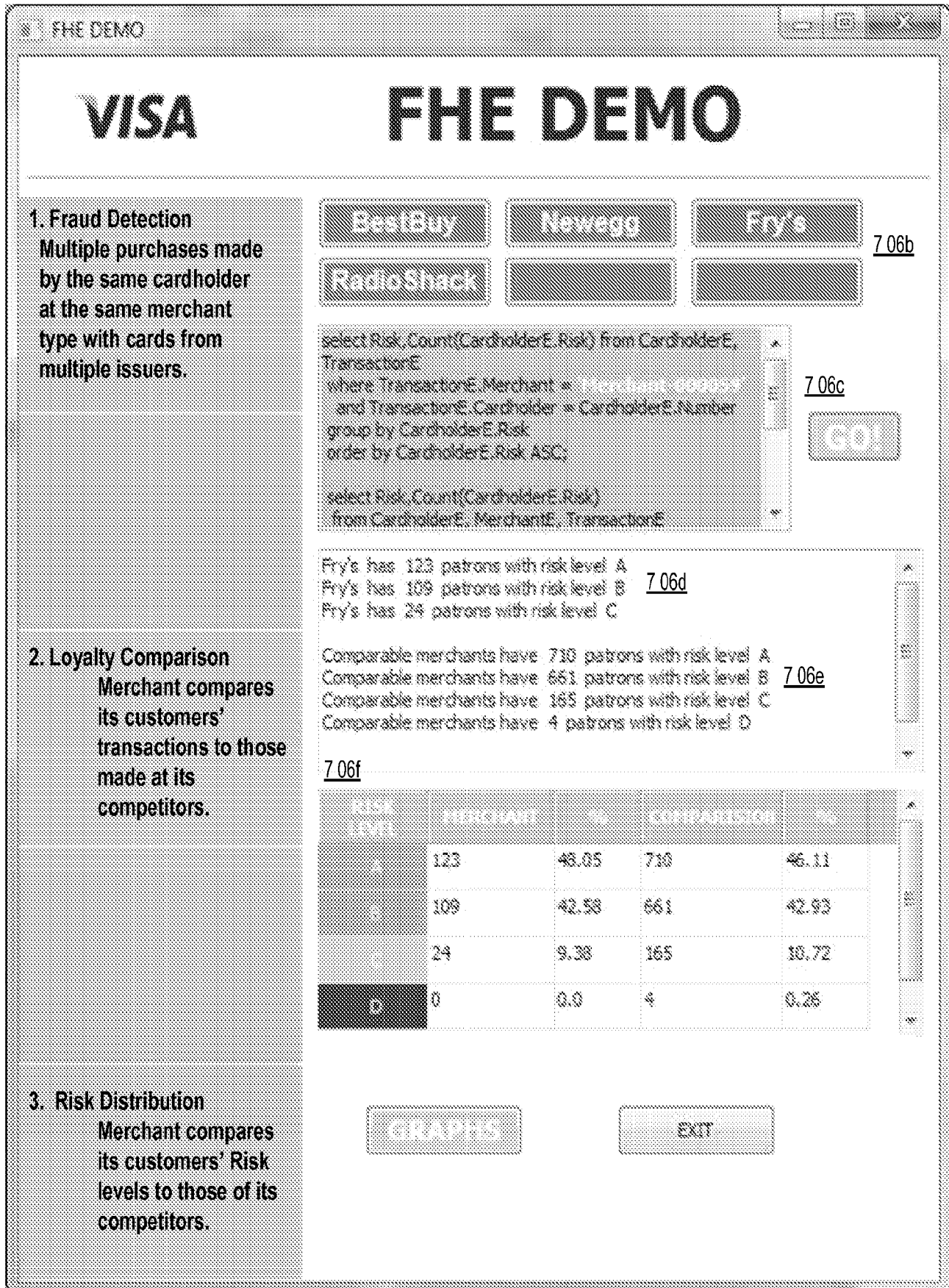


Figure 7F

Example: HEDO User Interface

707a

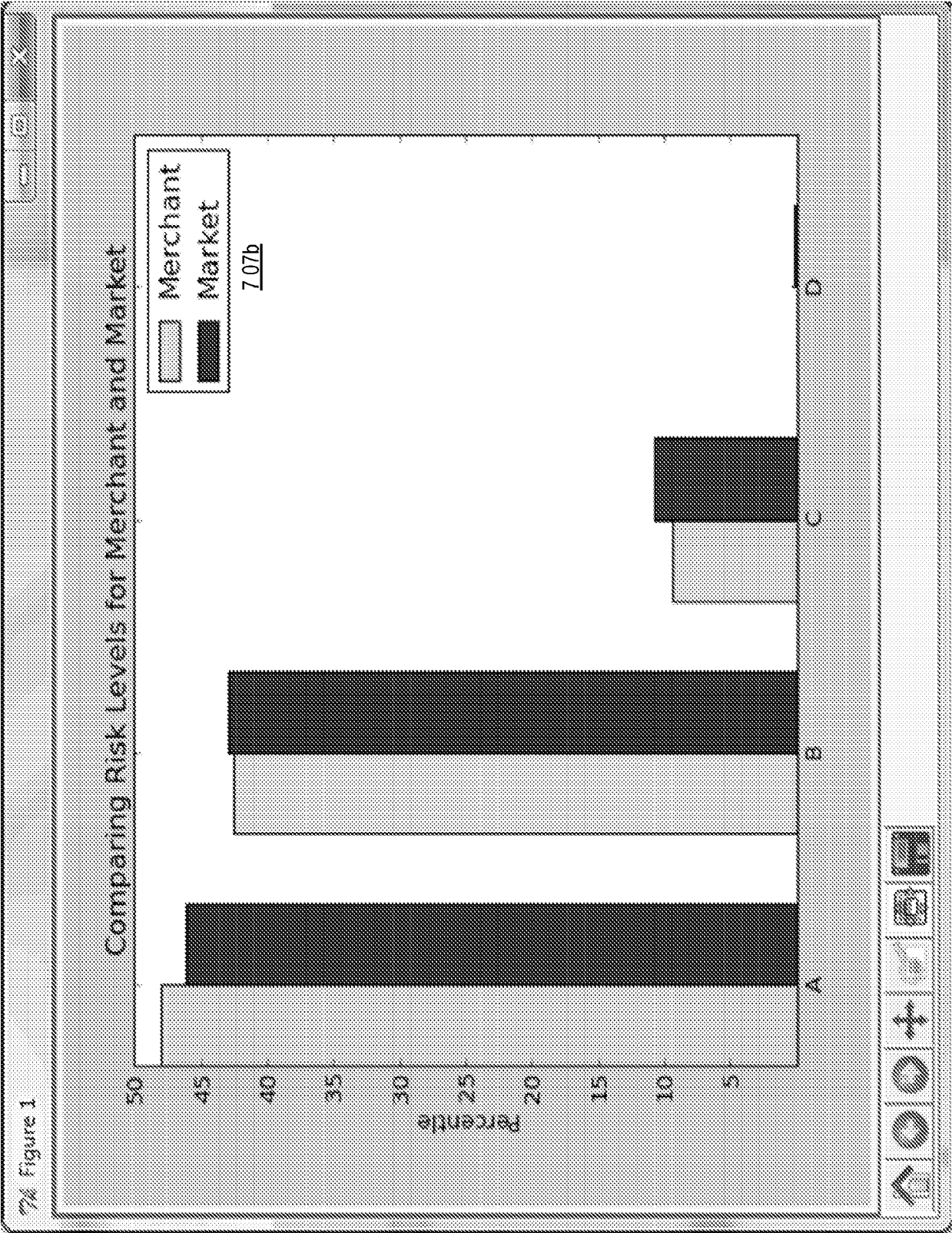


Figure 7G

Example: HEDO User Interface

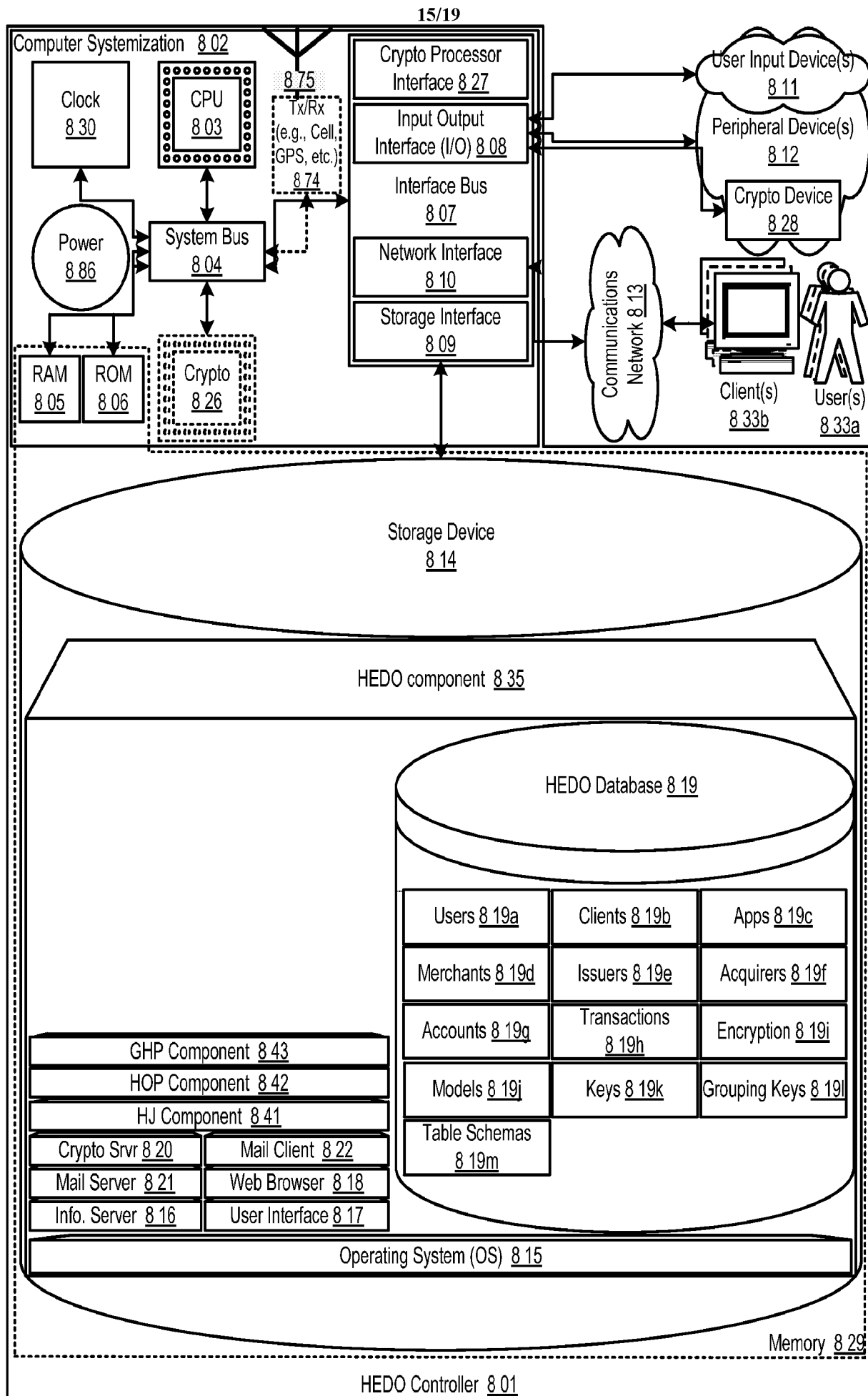


Figure 8

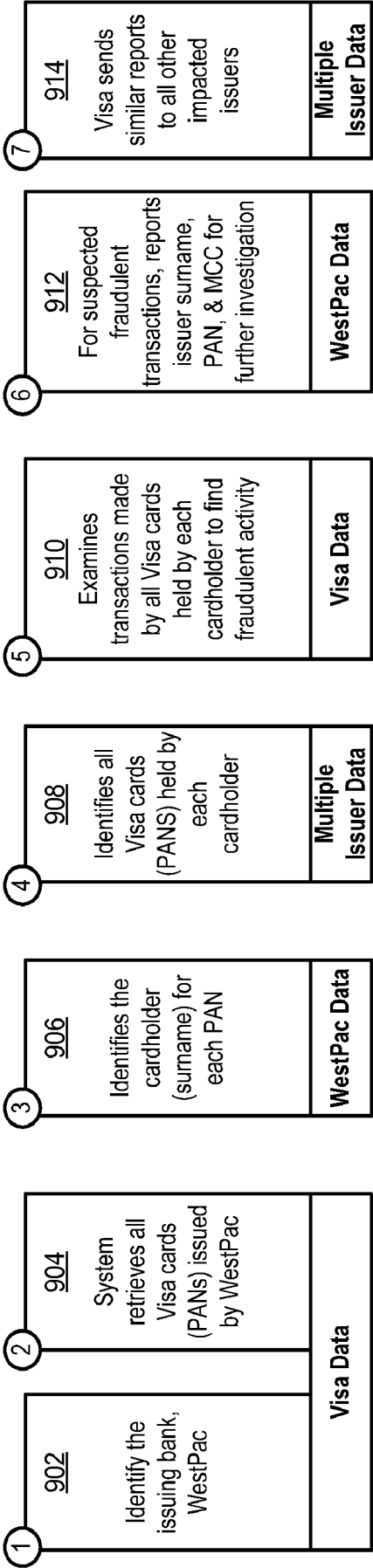


Figure 9

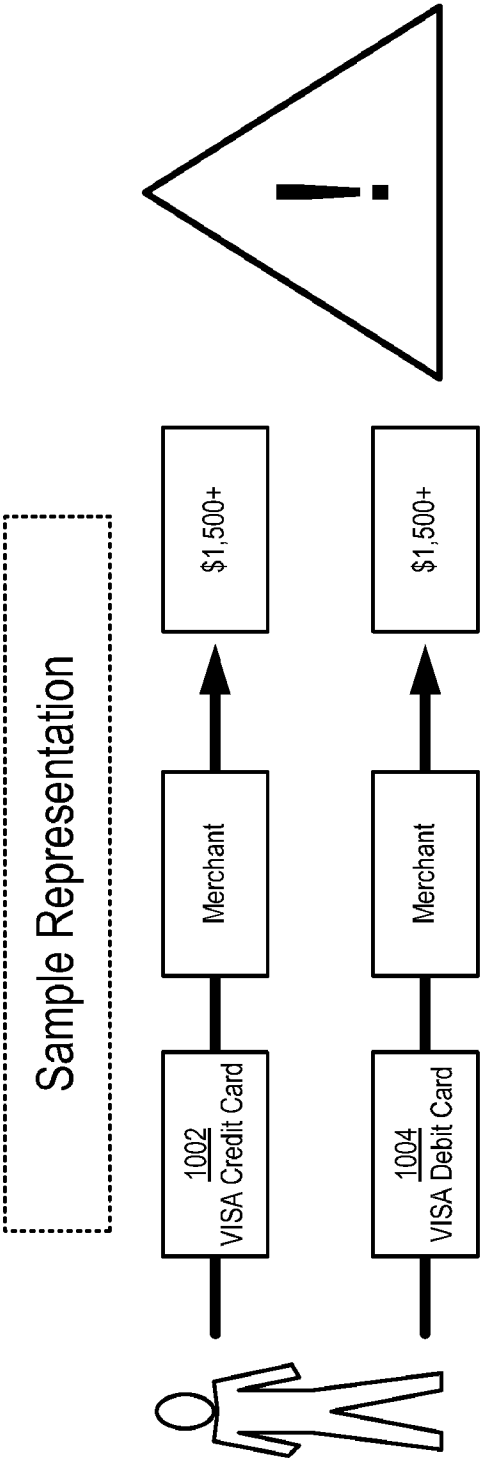


Figure 10

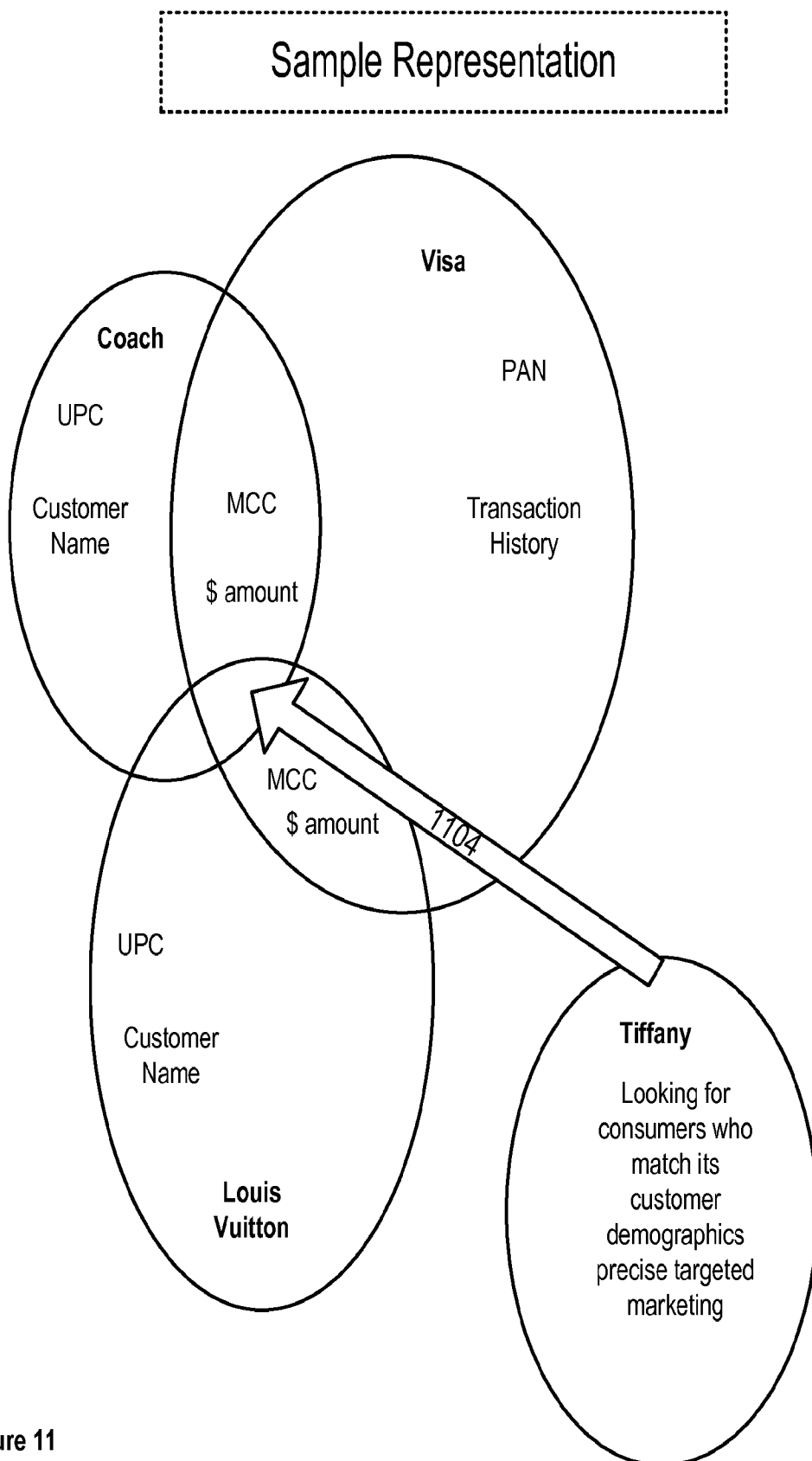


Figure 11

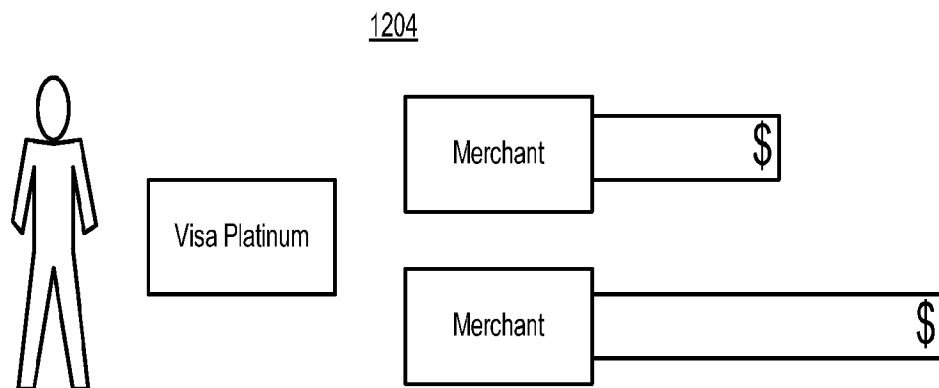
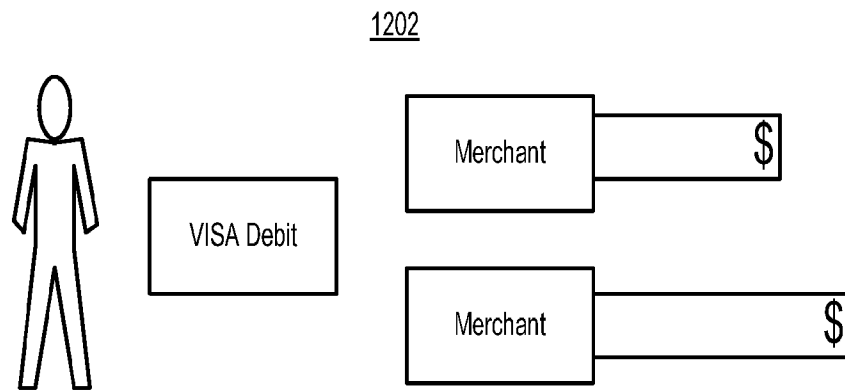


Figure 12