(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 December 2017 (21.12.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/216801 A1

(51) International Patent Classification:
G06F 21/60 (2013.01)

(21) International Application Number:
PCT/IL2017/050669

(22) International Filing Date:
15 June 2017 (15.06.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/350,738 16 June 2016 (16.06.2016) US

(71) Applicant: **BAR-ILAN UNIVERSITY** [IL/IL]; Ramat Gan, 5290002 Ramat Gan (IL).

(72) Inventors: **PINKAS, Binyamin**; 5 Brazil Street, Tel Aviv 6946015 (IL). **GEVA, Mordechai**; 6 Israel Yavin Street, Patach Tikva 4977389 (IL).

(74) Agent: **GASSNER, Dvir** et al.; 55 Yigal Alon Street, 6789115 Tel Aviv (IL).

(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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, 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, ST, 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:
— *with international search report (Art. 21(3))*

(54) Title: SECURE SHARING OF CONFIDENTIAL DIGITAL DATA

(57) Abstract: A system, method, and computer program product for secure comparison of confidential data items over a computer network. The system may include two or more parties, each possessing a separate collection of confidential data items. None of the parties to the comparison become exposed to any confidential data item possessed by any other party, or, at the very least does not become exposed to any whole confidential data item of another party.

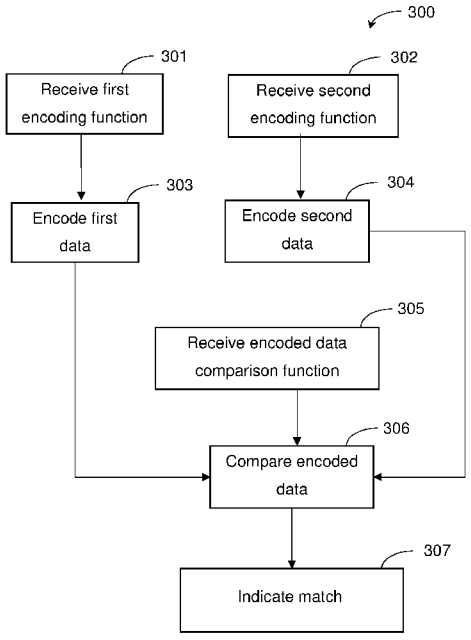

FIG. 3

WO 2017/216801 A1

SECURE SHARING OF CONFIDENTIAL DIGITAL DATA

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority from U.S. Provisional Patent Application No. 62/350,738, filed June 16th, 2016, entitled "SECURE SHARING OF SENSITIVE DIGITAL DATA".

FIELD OF THE INVENTION

[0002] The invention relates to the fields of cryptography and computer security.

BACKGROUND

[0003] Secure multi-party computation (also known as "secure computation" or "multi-party computation") is a subfield of cryptography with a goal of creating methods for parties to jointly compute a function over their inputs while keeping those inputs private.

[0004] Secure computation was formally introduced as secure two-party computation ("2PC") in 1982 by Andrew C. Yao, in "Protocols for Secure Computations", 23rd Annual Symposium on Foundations of Computer Science, 1982.

[0005] Yao introduced the "Millionaires' Problem", in which two millionaires may be interested in knowing which of them is richer without revealing their actual wealth. This problem may be analogous to a more general problem where there may be two numbers a and b and the goal is to solve the inequality $a \geq b$ without revealing the actual values of a and b.

[0006] A variant of the Millionaires' Problem is the "Socialist Millionaires' Problem", in which the two millionaires want to determine if their wealth is equal without disclosing any information about their riches to each other. This problem and its solution were first introduced by Markus Jakobsson and Moti Yung in "Proving Without Knowing: On Oblivious, Agnostic and Blindfolded Provers", Advances in Cryptology — CRYPTO '96, Volume 1109 of the series Lecture Notes in Computer Science, pp 186-200.

[0007] Both variants of the Millionaires' Problem may be important problems in cryptography and cyber security.

[0008] A prominent Yao-based function was implemented in 2004 in “Fairplay - Secure Two-Party Computation System” by Dahlia Malkhi, Noam Nisan, Benny Pinkas and Yaron Sella, published in USENIX Security Symposium 2004: 287-302. Fairplay comprises two main components. The first of these is a compiler enabling users to write programs in a simple high-level language, and output these programs in a Boolean circuit representation. The second component may then garble the circuit and execute a function to securely evaluate the garbled circuit.

[0009] Later, Y. Lindell and B. Pinkas, in "An efficient protocol for secure two-party computation in the presence of malicious adversaries," Eurocrypt 2007, vol. Springer LNCS 4515, pp. 52-78, 2007, introduced an efficient two-party computation function that is secure against active (malicious) adversaries. This technique was implemented by B. Pinkas, T. Schneider, N. Smart and S. Williams in "Secure two-party computation is practical," Asiacrypt 2009, vol. Springer LNCS 5912, pp. 250–267, 2009, which provided the first actively secure two-party evaluation of the Advanced Encryption Standard (AES) circuit, regarded as a highly complex (consisting of around 30,000 AND and XOR gates), non-trivial function (also with some potential applications), taking around 20 minutes to compute and requiring 160 circuits to obtain a 2-40 cheating probability.

[0010] The foregoing examples of the related art and limitations related therewith may be intended to be illustrative and not exclusive. Other limitations of the related art will become apparent to those of skill in the art upon a reading of the specification and a study of the figures.

SUMMARY

[0011] The following embodiments and aspects thereof are described and illustrated in conjunction with systems, tools and methods which are meant to be exemplary and illustrative, not limiting in scope.

[0012] One embodiment is a method for secure comparison of confidential data, the method comprising: computing an encoding of each item of a first list of confidential data

items, to produce a first set of encoded items; computing an encoding of each item of a second list of confidential data items, to produce a second set of encoded items; and comparing each of the encoded items of the first list against the encoded items of the second list, and indicating when a match is found, wherein, during execution of the method: a computer from which the first list has originated is not exposed to any whole one of the confidential data items of the second list, and a different computer, from which the second list has originated, is not exposed to any whole one of the confidential data items of the first list.

[0013] Another embodiment relates to a system for secure comparison of confidential data, comprising: a first computer configured to receive a first list of confidential digital data items; and a second computer configured to receive a second list of confidential digital data items, wherein: (i) the first and second computers are configured to communicate with each other over a computer network, (ii) either (a) the first computer or (b) the first and second computer jointly are configured to compute an encoding of each item of the first list, to produce a first set of encoded items, (iii) either (a) the second computer or (b) the first and second computer jointly are configured to compute an encoding of each item of the second list, to produce a second set of encoded items, (iv) either (a) the first computer, (b) the second computer, or (c) a third computer is configured to compare the encoded items of the first list against the encoded items of the second list, and to indicate when a match is found, and (v) during the computation and comparison operations, the first and third computers are not exposed to any whole one of the confidential data items of the second list, and the second and third computers are not exposed to any whole one of the confidential data items of the first list.

[0014] In some embodiments, the production of the encoded items of the first list is executed separately or independently of the production of the encoded items of the second list.

[0015] In some embodiments, the comparison of encoded items of the first list and encoded items of the second list is executed separately or independently of the production of encoded items from the confidential digital data items.

[0016] In some embodiments, the comparison of encoded items of the first list against the encoded items of the second list is executed on another computer, whereby the latter computer is not exposed to the confidential digital data items from either the first list or second list.

[0017] In some embodiments, the indication about a match may be indicated just to the first party, just to the second party or to both parties.

[0018] In some embodiments, the encoding of the digital data items of at least one of the first and second lists is based on a secure multi-party computation.

[0019] In some embodiments, the encoding of the digital data items of at least one of the first and second lists is based on a one-way function.

[0020] In some embodiments, the encoding of the digital data items of at least one of the first and second lists is based on a blind signature method.

[0021] In some embodiments, the encoding of the digital data items of at least one of the first and second lists is based on a Diffie-Hellman assumption.

[0022] In some embodiments, the encoding of the digital data items of at least one of the first and second lists is based on an oblivious evaluation of a function.

[0023] In some embodiments, the encoding of the digital data items of at least one of the first and second lists is based on an oblivious evaluation of a pseudo-random function.

[0024] In some embodiments, the encoding of the digital data items of at least one of the first and second lists is based on a k-wise independent function.

[0025] In some embodiments, the method further comprises at least one of adding, deleting, and updating digital data items in the first or second encoded items lists.

[0026] In some embodiments, the comparing of the encoded items is performed as a one-to-many comparison.

[0027] In some embodiments, the list of the confidential digital data items of the first list are maintained in the first computer and/or the list of the confidential digital data items of the second list are maintained in the second computer.

[0028] In some embodiments, the list of the secure encoding items of the first and/or second list are maintained in the first computer and/or in the second computer and/or in the third computer.

[0029] In some embodiments, the first or second computers automatically collects confidential items from other devices or networks connected to it.

[0030] In some embodiments, the comparison of encoded items of the first list against the encoded items of the second list is executed on a third computer, whereby the third computer is not exposed to the confidential digital data items from either the first list or second list.

[0031] In some embodiments, the indication about a match may be indicated just to the first computer, just to the second computer, or to both computers.

[0032] In some embodiments, each of the first and second computers is further configured for at least one of adding, deleting, and updating digital data items in the first or second encoded items lists, respectively.

[0033] Another embodiment provides a computer program product for secure comparison of confidential data, the computer program product comprising a non-transitory computer-readable storage medium having program code embodied therewith, the program code executable by at least one hardware processor to perform the steps of any one of the above embodiments of the method.

[0034] Another embodiment is a computerized method for secure comparison of confidential digital data values comprising using at least one hardware processor for performing the action of receiving, at each of a plurality of computers comprising a local confidential database, one set of a plurality of encoding processor instruction sets. Another action is computing, by each corresponding computer, an encoding of at least one value of the corresponding local confidential database according to the corresponding set of encoding processor instructions, to produce encoded database values on each corresponding computer. Another action is receiving at least some of the encoded database values and a set of encoded value comparison instructions. Another action is comparing the received encoded database values optionally using the set of encoded value comparison instructions to produce matching elements. Another action is issuing an indication of the matching elements.

[0035] In some embodiments, the receiving further comprises receiving a set of encoded value comparison instructions, wherein the set of encoded value comparison instructions comprises a comparison rule, and wherein the comparing is performed using the set of encoded value comparison instructions.

[0036] In some embodiments, the received encoded database values are compared with encoded database values derived from the local confidential database.

[0037] In some embodiments, the receiving and comparing are performed on one of the plurality of computers.

[0038] In some embodiments, the receiving and comparing are performed on a dedicated computer, wherein the dedicated computer is different from the plurality of computers.

[0039] Another embodiment is a computer program product for secure comparison of confidential data, the computer program product comprising a non-transitory computer-readable storage medium having program code embodied therewith, the program code executable by at least one hardware processor to perform the step of receiving, at each of a plurality of computers comprising a local confidential database, one set of a plurality of encoding processor instruction sets. Another step is computing, by each corresponding computer, an encoding of at least one value of the corresponding local confidential database according to the corresponding set of encoding processor instructions, to produce encoded database values on each corresponding computer. Another step is receiving at least some of the encoded database values and a set of encoded value comparison instructions. Another step is comparing the received encoded database values using the set of encoded value comparison instructions to produce matching elements. Another step is issuing an indication of the matching elements.

[0040] In some embodiments, the set of encoded value comparison instructions comprises a comparison rule.

[0041] In some embodiments, the received encoded database values are compared with encoded database values derived from the local confidential database.

[0042] In some embodiments, the receiving and comparing are performed on one of the plurality of computers.

[0043] In some embodiments, the receiving and comparing are performed on a dedicated computer, wherein the dedicated computer is different from the plurality of computers.

[0044] Another embodiment is a system for secure comparison of confidential data, comprising a plurality of computers, each comprising (i) a at least one hardware processor, (ii) a non-transitory computer-readable storage medium comprising a local confidential database, and (iii) a network connection. Each non-transitory computer-readable storage medium has program code embodied therewith, the program code executable by the respective at least one hardware processor to perform the step of receiving, at each of the plurality of computers, one set of a plurality of encoding processor instruction sets. Another step is computing, by each corresponding computer, an encoding of at least one value of the corresponding local confidential database according to the corresponding set of encoding processor instructions, to produce encoded database values on each corresponding computer. Another step is receiving at least some of the encoded database values and a set of encoded value comparison instructions. Another step is comparing the received encoded database values optionally using the set of encoded value comparison instructions to produce matching elements. Another step is issuing an indication of the matching elements.

[0045] In some embodiments, the set of encoded value comparison instructions comprises a comparison rule.

[0046] In some embodiments, the received encoded database values are compared with encoded database values derived from the local confidential database.

[0047] In some embodiments, the receiving and comparing are performed on one of the plurality of computers.

[0048] In some embodiments, the receiving and comparing are performed on a dedicated computer, wherein the dedicated computer is different from the plurality of computers.

[0049] In addition to the exemplary aspects and embodiments described above, further aspects and embodiments will become apparent by reference to the figures and by study of the following detailed description.

BRIEF DESCRIPTION OF THE FIGURES

[0050] Exemplary embodiments are illustrated in referenced figures. Dimensions of components and features shown in the figures are generally chosen for convenience and clarity of presentation and are not necessarily shown to scale. The figures are listed below.

[0051] FIG. 1 shows schematically a diagram of an exemplary system for secure multi-party computation;

[0052] FIG. 2 shows a flowchart of a method for secure multi-party computation; and

[0053] FIG. 3 shows a flowchart of method for secure multi-party computation with receiving functions.

DETAILED DESCRIPTION

[0054] Disclosed herein is a system, method, and computer program product for secure comparison of confidential data items over a computer network. Embodiments of the system may include two or more parties, each possessing a separate collection of confidential data items, such as data values.

[0055] According to present embodiments, the confidential data items of all parties undergo an encoding process, and the comparison may be made between *encoded* items. This way, the comparer (which may be the Producer, the Consumer, or even an independent third party) is not exposed to the confidential items themselves – but to their encoded versions. This makes the comparison *secure*. According to another embodiment, the comparison function comprises a comparison rule analytically convolved with the decoding function so that the comparison may be made directly on the encoded data.

[0056] For example, by providing an encoding function to a first party and a corresponding encoding function to a second party, the first party may encode a first confidential data values known to the first party, and send the encoded first confidential data values to the second party. The second party may use the corresponding encoding function to encode second confidential data, and an encoded data comparison function, comprising a comparison rule, to determine if encoded items from the second confidential data values comply with encoded items from the first confidential data according to the comparison rule.

[0057] For example, each of the email addresses of the crime suspects is encoded by a first party, and so are each of the email addresses logged by the servers of a second party. The encoding may be impractical to reverse by the second party or the third-party comparer, thereby maintaining the confidentiality of the data items during the comparison. The comparison itself, on the other hand, may be computationally simple, such that it may be conducted rapidly for a large number of encoded items.

[0058] For reasons of simplicity, the following description relates to a scenario in which two parties compare their confidential data items. However, the techniques disclosed herein may be applicable, without the need for undue experimentation, to a multi-party comparison involving more than two parties, each with same of different collections of data.

[0059] The term "secure", as used herein to describe the comparison, denotes that none of the parties to the comparison becomes exposed to confidential data items possessed by any other party. A similar term, "partially secure", refers to when each party may be exposed to *some* confidential data item of another party, such as exposed to *a portion* of one or more confidential data item, some but not all confidential data items, *a portion* of certain information, such as metadata, about one or more confidential data items. Both "secure" and "partially secure", characterize the comparison between data items of the parties may be a process which maintains the confidentiality of each party's confidential data items versus the other party or parties, versus one or more third parties, and/or the like.

[0060] Reference is now made to FIG. 1, which shows schematically a diagram of an exemplary system 100 for secure multi-party computation. A first computer 100 comprises at least one hardware processor 101, a non-transitory computer-readable storage medium 102, a network interface 103, and a user interface 104. Non-transitory computer-readable storage medium 102 comprises a first confidential database 102A, such as a list of data values, a data encoder 102B module, and optionally a data comparer 102C module. Network interface 103 is connected to an Internet 130. A second computer 110 comprises at least one hardware processor 111, a non-transitory computer-readable storage medium 112, a network interface 113, and a user interface 114. Non-transitory computer-readable storage medium 112 comprises a second confidential database 112A, such as a list of data values, a data encoder 112B module, and optionally a data comparer 112C module. Network interface

113 is connected to Internet 130. One or more external comparers 121, 122, and 123 may be connected to Internet 130 for comparing the encoded values from the first computer with the second computer, or the comparing may be performed by Encoder 102B and/or Encoder 112B. The comparison results may be indicated to a user on user interface 104 and/or 114.

[0061] Reference is now made to FIG. 2, which shows a flowchart of a method 200 for secure multi-party computation. Flowchart 200 comprises encoding 201, 202, and 203 a first, second, and/or third confidential data, each on separate computers, and comparing 204 the encoded data on one of the separate computers or a different computer. The results of the comparison may be indicated 205 as the secure multi-party computation.

[0062] Reference is now made to FIG. 3, which shows a flowchart of method 300 for secure multi-party computation with receiving functions. Method 300 may comprise receiving 301 and 302 at each of a first and second computer an encoding function, and using the respective encoding function to encode 303 and 304 confidential data on each computer. Method may comprise receiving 305 an encoded data comparison function, and comparing 306 the encoded data. The results of the comparison may be indicated 307 as the result of the secure multi-party computation.

[0063] To better illustrate the invention, the following description is interwoven with an exemplary scenario in which one of the parties is called a "Producer" and the other a "Consumer". For example, the "Producer" is a law enforcement agency possessing a collection (or a "list") of email addresses of crime suspects. These may be the confidential data items of the Producer. The Producer wishes to query a certain internet service provider (ISP), the "Consumer", on whether the communication it channels through its servers includes email messages relayed to or from the email addresses of the crime suspects. When such email messages are found, the Producer may request the Consumer to provide their contents, for example after acquiring a proper warrant. However, for obvious reasons, the Producer may not simply circulate the list of email addresses to the Consumer – these email addresses may be confidential, and must remain confidential – either according to the law or simply in order to preserve law enforcement interests, counterintelligence, and/or the like. The confidentiality may be breached if ISP employees may be exposed to the actual email addresses flagged by law enforcement to be associated with crime suspect.

[0064] A similar sensitivity concern arises also with respect to the communications channeled and logged by the Consumer, such as the ISP. Privacy and other laws often prevent public service providers from sharing private information of their clients with third parties, including with law enforcement agencies that may not produce a suitable warrant. Accordingly, in this example scenario, the Consumer regards the email-related communications it channels as confidential, and would desire to keep them confidential from the Producer. Namely, the Consumer may not agree to simply share all email addresses (referred to below for convenience as "input items") logged by its servers with the Producer.

[0065] Embodiments of the present invention solve problems arising in multidirectional confidentiality conflicts, allowing the Producer and the Consumer to compare their confidential items without any of these items being exposed to the opposing party – at least not exposure of an *entire* data item. This may solve legal and/or business confidentiality concerns.

[0066] In some embodiments, each party computes the encoding of its own data items, while in some other embodiments – both parties engage in a secure computation of the encoding according to secure computation functions known in the art.

[0067] When the comparison of a certain encoded item of the Producer with a certain encoded item of the Consumer indicates a match, one or both parties may be alerted. For example, the Producer may be alerted, so that it may seek a warrant ordering the Consumer to produce all email messages relayed to or from the email address underlying the encoded item that was matched. In other situations, the mere indication of a match may be a sufficient result for the parties involved.

[0068] In some embodiments, the comparison may take place between more than one encoded item of each party at a time, such as between two encoded items of the Producer and two encoded items of the Producer, or even more.

[0069] In some embodiments, the comparison may be a binary computation (namely, either a match may be found or not), while in some other embodiments, the comparison may be a statistical computation that yields a non-binary likelihood or similarity score of how strongly the encoded items are matched. In the latter case, a probabilistic data structure may be used to test whether an encoded data item may be part of a set of mutually-encoded data

items. An example of a suitable technique is the Bloom filter, in which false positive matches may be possible, but false negatives are not.

[0070] Reference is now made to Fig. 1, which shows a network diagram of an exemplary, simplistic, embodiment. A first computer and a second computer, each encompassing a collection of confidential data items, communicate over a computer network (such as the internet), and compare the encoded versions of these confidential data items.

[0071] In terms of physical components making up the system of the present embodiments, these may include one or more computers for each of the Producer and the Consumer, and a computer network interconnecting the computer(s) of the Producer and the computer(s) of the Consumer. Each computer may include one or more Central Processing Units (CPUs, or "processors" for short), a volatile memory such as Random-Access Memory (RAM), and a non-volatile memory such as one or more optical, magnetic, or flash disks that store the confidential data items and/or their encodings. Additionally, or alternatively, one or more of the computer(s) may be embodied in a Field-Programmable Gate Array (FPGA) chip or an Application Specific Integrated Circuit (ASIC) chip.

[0072] One or more of the parties may employ an item sensor – a software or hardware component that automatically collects confidential data items to be encoded. For example, the ISP Consumer may operate a software and/or hardware agent that monitors data traffic passing through the ISP's servers and extracts email addresses appearing in the traffic.

[0073] The present system may facilitate complex relationships between multiple Producers and multiple Consumers, wherein a certain comparison session may involve more than one Consumer and/or more than one Producer. Similarly, the present system may facilitate the existence of multiple comparers that are not the Producers or the Consumers.

[0074] Optionally, each part of the system, such as a Producer sub-system or a Consumer sub-system, may include a user interface (UI) and/or an Application Programming Interface (API) which allow the parties to manually or automatically upload their confidential items for the purposes of encoding and/or sending the encoded items for comparison. Additionally, an API and/or UI may be embodied in a central server, which the parties may manually or automatically upload their encoded items for comparison – in case that server is the comparer.

[0075] Alternatively, the present system may be of a distributed nature, utilizing the computerized resources of the Producer and the Consumer. In such distributed system, there may still be an API at each party's end for enabling the other party to transmit data in a convenient and standardized manner. There may also be an API and/or UI at a certain party for use by the party itself – for example in order to feed confidential data items from other systems of the party to the present system.

[0076] Present embodiments may possess the following advantageous properties:

A. Online adaptivity – changes to the lists of confidential data items of both parties, such as adding, deleting or updating an item, may be made on the fly.

B. Efficiency and Scalability – the system may support very large collections of confidential data items from both parties, and have a high throughput.

C. Real-time responsiveness – the system may complete the following functions in low latency: updating the Producer list, and answering Consumer queries.

[0077] A major challenge in constructing the present privacy-preserving data sharing system, may be to support the concerns that were described above, while preserving privacy. The approach taken by present embodiments is to encode and compare items on a *single-item* basis (or on the basis of a low number of items every time), such that the lists of confidential data items on both sides may be quickly and efficiently updated, without having to wait for accumulation of many items on a certain list before comparing. The present approach may therefore be different than the older, Private Set Intersection (PSI) approach, which necessitated whole sets of items to be compared, instead of individual items. This difference is discussed later in this specification.

[0078] The present approach, as briefly discussed above, may compute a secure encoding of each item in the list of confidential items. In most (but not all) embodiments of this approach, the computation may be done using a function that may be run by the Producer with the help of the Consumer. The parties therefore run a joint computation where the Producer (or comparer) learns the secure encodings of the items in the confidential list. In addition, when the Consumer receives an input item, it sends the secure encoding (which it may compute by itself, or otherwise together with the Producer) of this item to the Producer

(or comparer). The Producer (or comparer) may then compare this value to the secure encodings of the confidential items, and check when there is a match.

[0079] A variant of this approach may be to compute a secure encoding of each of the confidential items of the Consumer, using a function that may be run with the help of the Producer. The Producer sends to the Consumer the list of secure encodings of the confidential items. In addition, the Consumer and the Producer run a process in which the Consumer learns the secure encodings of its items. These encodings may then be compared with the secure encodings of the confidential items of the Producer. The drawback of this approach may be that it requires computing the secure encoding of each confidential input item of the Consumer, rather than of each confidential item of the Producer. In many settings, the number of input items of the Consumer may be far greater than the number of confidential items of the Producer and therefore this approach will be less efficient than the previous one (computing secure encodings of the confidential items of the Producer). Thus, every situation may have its own better-suited embodiment.

[0080] The parties may compute a secure encoding of values, which may be defined as follows:

[0081] Secure Encoding – An encoding of a confidential items, optionally created using additional secret functions (e.g. secret keys). The secure encoding does not reveal to the entity that holds the encoded it the value of the confidential item (or at least not the entirety of the confidential item) or the encoding functions that were used to produce the secure encoding. Without loss of generality, a secure encoding may be computed using one-way-functions, hash functions, encryption functions, secure two-party computation, secure multi-party computation, or other techniques. A detailed description of different methods for computing secure encodings appears later in the specification.

[0082] Procedure for adding an item:

A. Producer receives a confidential item(s) to add through one or more of its input channels.

B. Producer, alone or with Consumer, with additional parties, and/or the like, pre-processes item value(s) to a produce a secure encoding. Namely, this computation may be done by the Producer which computes, for example, a hash of the item; or the computation may be

jointly done by the Producer and the Consumer, possibly by performing a secure computation.

C. The result of (B) may be received at the Producer, Consumer or Comparer.

[0083] Procedure for searching input:

A. Consumer receives a confidential item(s) through one or more of its input channels.

B. Consumer, alone or together with Producer, or with additional parties, pre-processes item(s) to a secure encoding. Namely, this computation may be done by the Consumer alone by computing, for example, a hash of the item; or the computation may be jointly done by the Producer and the Consumer, possibly by performing a secure computation.

C. The result of (B) may be received at the Producer, Consumer or Comparer.

D. The Comparer tests if the secure encoding matches or alternatively resembles an item in its list.

E. When a match or similarity is found, the Comparer reports the match or similarity to the Producer, Consumer, or both.

[0084] Procedure for adding, deleting or updating an item:

A. Producer receives through one or more of its input channels a confidential item(s) to add, delete or update.

B. Producer, alone or with Consumer, pre-processes added or updated item(s) to a secure encoding. Namely, this computation may be done by the Producer alone which computes, for example, a hash of the item; or the computation may be jointly done by the Producer and the Consumer, possibly by performing a secure computation. In case a deletion was instructed, both the confidential item and its encodings may be simply deleted.

C. The result of (B) may be received at the Producer, Consumer or Comparer.

[0085] Provided herein are a number of exemplary techniques for computing a secure encoding. Those of skill in the art will recognize that the present system may utilize other techniques which are not described here.

[0086] The Producer receives an input data vale denoted X and computes the secure encoding SC(X) of X. The computation may be dependent on a secret data (key) that may be known to the Consumer. (The setting might be different, with the secret encoding based on a key known to the Producer, and computed over an input given by the Consumer, but we describe just a single setting here in order to keep the description simple.) The computation may be based on a function that may be secure against semi-honest adversaries, against malicious adversaries, against covert adversaries, and/or the like.

[0087] Listed below are exemplary techniques for computing a secure encoding:

A. Using secure two-party or multi-party computation.

B. Using a one-way function.

C. Using operations similar to blind signatures.

D. Using operations based on the Diffie-Hellman assumption.

E. Computing an oblivious pseudo-random function (PRF).

F. Using oblivious polynomials or other k-wise independent functions.

[0088] These methods, which may be further discussed below, enable to efficiently create a secure encoding of confidential items and inputs for both Producer and Consumer, thereby enabling to adaptively and efficiently add, delete or update a list, as well as to test the existence of a secure encoding of a Consumer's input in a list of Producer's confidential items (i.e. to "compare"), and to allow real-time responsiveness for the system.

[0089] When using secure two-party or multi-party computation, the computation of the secure encoding of X may depend on a secret key K which may be known to the Consumer. There may be a function denoted A, such that SC(X) = A(K,X). That is, the function receives both K and X as inputs. The function A may compute a function which is at least a one-way. For example, it may be the computation of an encryption of X using an encryption algorithm that uses K as a key (for example, using the AES encryption algorithm). Namely, A(K,X)=AES(K,X). It may also be a keyed hash function of X, keyed with A, for example A(K,X) = H(K || X), where H denotes a hash function such as SHA256 or similar functions, and K and X may be concatenated and used as an input to H (the concatenation may be in any order or in any other type of mixing of the inputs).

[0090] The computation may be done using previously methods for secure two-party computation. This may be a well-known technique for enabling two parties to compute any function of their private inputs while hiding everything about their inputs except for the output of the function. There are well known functions for secure two-party computation, for example the Yao function which was extensively investigated and optimized, as briefly discussed in the Background section above. This secure computation may be run very efficiently and ensure that the Producer learns A(K,X) and nothing else, and the Consumer learns nothing.

[0091] Optionally, the secure computation may be assisted by additional parties that are involved in the computation, typically in order to further improve performance. In this case it may be possible to use methods of secure multi-party computation (this term refers here to computations involving more than two parties).

[0092] The secure encoding may be the output of a one-way function. A typical example for such functions is a hash function which is not keyed. Namely, SC(X)=H(X). In this case, the Producer may compute the secure encoding by itself, without any collaboration with the Consumer. The drawback in this solution may be that when the Consumer sends to the Producer the secure encodings of its input items, the Producer may attempt to run a dictionary search or brute force search and attempt to identify the input items. Namely, suppose that for an input item Y that the Consumer sends to the Producer the value H(Y). Suppose also that inputs come from a domain D which may be small or has small min-entropy. Then the Producer may search over the possible, or the most likely, values Z in D, compute H(Z) for each such item, and look for an item Z for which it holds that H(Z)=H(Y). This item may be likely to be equal to Y. Therefore, the option of using a non-keyed hash function for the secure encoding may be used when this type of attack may not be possible, namely when the domain D has a large min-entropy.

[0093] Another option may be based on blind signatures, which are typically based on the RSA function but may also be based on other techniques. In a blind signature, a signer has a signing key K. A user has an input X. The two parties run a function where the user learns the signature of X, without learning the signature key, and the signer learns nothing about

X. The first such function was suggested by Chaum based on the RSA assumption, and there may be other well-known solutions for this problem. See Chaum, David (1983). "Blind signatures for untraceable payments" (PDF). Advances in Cryptology Proceedings of Crypto 82 (3): 199–203.

[0094] For our purposes, the Consumer may be the signer, and the Producer may be the user. The confidential item X denotes the input, and the Producer obtains the signature on this item. In addition, the Consumer sends to the Producer (or Comparer) the signatures on its input items, and the Producer (or Comparer) may compare them to the signatures that it obtained (blindly) of the confidential items.

[0095] Other methods for computing blind signatures may also be possible. Also, there may be no need to compute and send the entire signatures. Instead, some of these values may be replaced with hashes of signatures, or with other functions that may be computed using a key which may be known to the Consumer.

[0096] Another option may be based on the hardness of computing discrete logarithms in some groups, and on the Diffie-Hellman security assumption which may be related to computing discrete logarithms. The Consumer has a secret key K, and the Producer has a function where the Producer that has an input X learns the value $(H(X))^K$, without the Producer learning anything about K and without the Consumer learning anything about X. The function H denotes a hash function. The exponentiation may be done in a group where the Diffie-Hellman assumption holds and where it may be hard to compute discrete logarithms. Afterwards, the Consumer may send to the Producer the value $(H(Y))^K$ for each input Y that the Consumer wants to compare to the suspects lists. H(X) denotes a function that maps X, which may be outside the group, into the group; for example, using a hash function to map $\{0,1\}^* \rightarrow \{0,1\}^t$. Note that in some cases H(X) may be an arbitrary function, including the identity function.

[0097] This computation, and known and straightforward variants thereof, may be run in any group in which the relevant assumptions hold, for example in the group of integers modulo a prime number, or in elliptic curve groups.

[0098] Another option may be based on computing an oblivious pseudo-random function evaluation. In this setting, the Consumer has the key to a pseudo-random function F, and

the Producer has an input X. The computation function lets the Producer learn F(K,X) without learning K and without leaking any information to the Consumer.

[0099] An example of a pseudo-random function for which there may be an efficient function for oblivious evaluation is the Naor-Reingold pseudo-random function (Naor, M., Reingold, O. "Number-theoretic constructions of efficient pseudo-random functions," Proc 38th IEEE Symp. on Foundations of Comp. Sci, (1997), 458-467), and other functions that were described in M. Freedman, Y. Ishai, B. Pinkas and O. Reingold, "Keyword Search and Oblivious Pseudorandom Functions", Proceedings of 2nd Theory of Cryptography Conference (TCC '05) Cambridge, MA, Feb 2005, and in subsequent work.

[00100] Another option may be to compute the secure encoding as the output of a k-wise independent function. Such functions have the property that any k outputs of the function may be distributed independently of each other. An example of such a function is a polynomial of degree k-1. In this setting, the Consumer knows the description of a k-wise independent function, and the Producer has a confidential item X. The two parties run a function in which the Producer learns the output of the function on X, and no other information, and the Consumer learns nothing about X. If the Producer learns at most k-1 such values, then it may not deduce anything about any other value of the function. Such functions, in particular polynomials, were used for private set intersection, for example by M. Freedman, K. Nissim and B. Pinkas, "Efficient Private Matching and Set Intersection", Advances in Cryptology - Eurocrypt '2004 Proceedings, LNCS 3027, Springer-Verlag, pp. 1-19, May 2004, or by Lea Kissner, Dawn Song, "Privacy-Preserving Set Operations", CRYPTO 2005.

[00101] We note however, that one must be very careful in using such functions, since if the server knows some potential inputs of the Consumer (for which the Consumer sends the output of the function) then it might know overall more than k-1 values and be able to learn some information. We also note that hashing schemes (in particular, hashing into bins) may be used to map different input values to be computed with different functions.

[00102] As briefly discussed above, the approach taken by present embodiments may be superior to the previous PSI approach for the scenarios the present embodiments may be intended.

[00103] There may be several parameters by which a comparison between the present embodiments and the PSI approach may be made. For example:

A. Resource consumption in terms of memory, CPU utilization, and bandwidth.

B. Time to calculate the intersection return a result.

C. Complexity of adding new items from either side, i.e. Producer or Consumer.

[00104] These parameters may generally define whether the system is fast enough, whether it is efficient and scalable, and when it may provide answers in real-time.

[00105] To date, in order to solve an information sharing problem as described herein, one may typically use a Private-Set-Intersection (PSI) function, which finds the intersection of two sets that may be respectively known to two parties, without disclosing to any of the parties any other information about the set of the other party.

[00106] A comparison of different state-of-the-art methods for computing PSI appears in Benny Pinkas, Thomas Schnedier and Michael Zohner, "Private Set Intersection based on OT Extension", Usenix Security '2014. An efficient method for computing PSI is described in Benny Pinkas, Thomas Schnedier, Gil Segev and Michael Zohner, "Phasing: Private Set Intersection Using Permutation-based Hashing", Usenix Security '2015.

[00107] According to Pinkas-Schneider-Zohner 2014 and Pinkas-Schneider-Segev-Zohner 2015, most PSI functions may be in one of these categories:

A. PSI functions based on public key operations, such as exponentiations, and whose security is based on the DH assumption and the hardness of computing discrete logarithms, or on the RSA assumption and the hardness of factoring large numbers. According to Pinkas-Schneider-Segev-Zohner 2015, the most efficient such function, based on DH over elliptic curve groups, runs, over a fast LAN network, in 818 seconds when both sets may be of size 10^6 items, or in 422 seconds when one set may be of size 4000 and the other of size 10^6 items.

B. PSI functions based on generic secure computation techniques that represent the PSI functionality as a circuit. According to in Pinkas-Schneider-Zohner 2014, the best such function runs, for sets of size 256,000 items, over a fast LAN network, may be about 762

seconds (for security level of 128 bits, which may be the security level we use for our following comparisons).

C. PSI functions based on oblivious transfer. According to Pinkas-Schneider-Segev-Zohner 2015, the most efficient such function, runs, over a fast LAN network, in 13.5sec when both sets may be of size 10^6 items, or in 7sec when one set may be of size 4000 and the other of size 10^6 items.

D. PSI functions based on using additional trusted parties. These functions may not be applicable in setting where there are no such parties that may be trusted.

[00108] From these performance numbers, PSI functions may not be suited for handling a large number of queries, or for providing answers in real-time, or for handling a large set size.

[00109] The main problem with methods for computing PSI may be that *each* computation of PSI may require running a number of cryptographic computations which may be linear, or perhaps even quadratic, in the size of the sets whose intersection may be computed. By "cryptographic computations" we refer to operations such as public key operations, exponentiations, oblivious transfers, or even symmetric key operations (such as AES), which were described in the cryptographic literature. The limiting issue with this type of computation may be that they typically have a relatively high computation overhead. Therefore, doing a large number of these computations per comparison may be prohibitive in terms of performance. Namely, in order to support high performance, the number of cryptographic operations per comparison should not depend on the total size of the sets, but rather on the size of the current query. Another problem with computing PSI may be that the communication exchange between the parties may be at least linear in the number of inputs.

[00110] When using PSI based functions for the purpose of performing information sharing, it may be used to compute the intersection between the set of confidential items known to the Producer, and a set of input items known to the Consumer. (The size of this latter set depends on the frequency with which the Consumer checks for intersections. It could consist of a single item – if the Consumer prefers to immediately query about each input item that arrives, or of multiple items if the Consumer prefers to query about batches

of input items.) Therefore, computing PSI may require computing a large number of cryptographic operations per comparison, which might be too inefficient. Furthermore, in order to overcome this performance issues with using PSI, one might attempt to compute PSI on batches of inputs. Namely, have the Consumer wait and compute the intersection after it obtained a batch of multiple input items. However, the usage of this latter approach does not enable to check Consumer inputs in real-time, since it essentially defines a tradeoff between the amortized computation effort per input item and the latency in obtaining answers about input items. Additionally, the *size* of the set of confidential items may become too large over time, making each comparison take longer and longer, and may require more and more computational, memory, and/or bandwidth resources; ultimately such an intersection eventually becomes too big to compare.

[00111] In addition to the low performance of PSI functions, they suffer from more problems preventing their usage in information sharing applications:

A. Security against malicious adversarial behavior: The most efficient PSI functions may be secure against adversaries that may be known as "semi-honest", or "honest but curious". Such adversaries may be trusted to follow the instructions given to them by the function that they may be asked to follow. A much stronger type of adversaries may be denoted as "malicious", and may be assumed to behave arbitrarily (not necessarily following the function). It may be preferable of course, and in some cases inevitable, to run a computation that may be secure against malicious adversaries. However, PSI functions with this level of security may be even less efficient (and considerably so) than the PSI functions that we described. On the other hand, the solutions that we describe may provide security against malicious adversaries.

B. Updates: Our solutions enable to continuously add input items, both to the input list of the Consumer and to the list of confidential items of the Producer. The overhead of adding an item does not depend on the number of existing items (by overhead we refer to the number of cryptographic operations that need to be computed, the memory requirements and bandwidth). On the other hand, when new items may be added then PSI solutions may be re-calculated and applied to previous items of the items (in addition to the new items).

Therefore, PSI functions may be much less efficient in handling updates to the input data sets.

[00112] The functions of the present embodiments use very few cryptographic operations per query. Therefore, they use much fewer resources and obtain both a small computation effort per input item, and very low latency for obtaining answers, relative to PSI based functions.

[00113] The present system comprises a Producer that may continuously add new items to a list, without the need to recalculate the entire intersection. Similarly, and potentially even more effectively, the Consumer may continuously and in real-time test new items without the need to recalculate the entire intersection. Testing of whether a secure encoding appears in the list may be done as simply as a trivial comparison, which implies a high degree of efficiency and scalability.

[00114] Per the following potential implementations, the estimated resources for creating a secure encoding by the Producer and Consumer may be detailed below:

[00115] Using secure two-party or multi-party computation. We assume, for example, that the secure computation may be of the AES encryption function which may be used as a pseudo-random function. A circuit for computing AES has approximately 33,000 gates. In comparison, a circuit for computing PSI of sets of size N has, according to Pinkas-Schneider-Zohner 2014 about $3 \times s \times N \times \log N$ gates, where S denotes the length of the inputs. As an example, consider the case of S=128 and $N=10^6$, in which case the number of gates may be about 7.2×10^9 gates. For S=32 and $N=10^4$ the number of gates may be about 1.2×10^7. In both cases this number may be larger by many orders of magnitude than the number of gates for computing an AES circuit.

[00116] For a non-keyed hash, such as SHA-256, the one-way function may be easily and efficiently computed independently on each side, and may not require any communication or joint computation in order to create the secure encoding. As described above, this configuration applies on large domains, whereby brute-force, or dictionary based attacks or similar may not be executed efficiently.

[00117] Using operations similar to blind signatures, the computation may require computing a function similar to the RSA function. The throughput for computing this function may be typically a few hundreds or thousands of computations per second.

[00118] Using operations based on the Diffie-Hellman assumption, may require each side to compute exponentiations on appropriate groups. Performance may be improved when using efficient implementations over elliptic curve groups, and execution on high performance GPUs (Graphic Processing Units). On a regular CPU, ECC (Elliptic Curve Cryptography) operations may reach several tens of thousands of operations per second (e.g. D. J. Bernstein, Tanja Lange, T. U. Eindhoven, Peter Schwabe, "The security impact of a new cryptographic library", Web, https://cr.yp.to/talks/2011.09.28/slides.pdf, last viewed June 11, 2016).

[00119] Computing an oblivious pseudo-random function (PRF). When computing the Naor-Reingold function, the computation may require a single exponentiation and multiple multiplications (which may be more efficient than exponentiations). The overhead may therefore be worse than that of the Diffie-Hellman based constructions, but not by much.

[00120] Using oblivious polynomials or other k-wise independent functions. This approach may require computing k exponentiations, or alternatively compute a polynomial over a field of size S by doing log(S) oblivious transfers (which may be efficiently implemented using oblivious transfer extension), and may require sending at least k data items per query. However, this approach may be used for up to k queries before a new polynomial of function may be used, and may therefore be limited in its usage.

[00121] By using extremely fast cryptographic primitives such as, but not limited to, the ones included in the present embodiments, the present system enables orders of magnitude better and faster secure computation operations. Using the present system, private information sharing may take place in real-time, and allow to adaptively add and remove private items on either Producer or Consumer sides, in an efficient and scalable manner, and may be applied in real-world systems with big-data scale, and presents a significant advance compared to prior art, such as PSI.

[00122] The present invention may be a system, a method, and/or a computer program product. The computer program product may include a computer readable storage medium

(or media) having computer readable program instructions thereon for causing a processor to carry out aspects of the present invention.

[00123] The computer readable storage medium may be a tangible device that may retain and store instructions for use by an instruction execution device. The computer readable storage medium may be, for example, but not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing. A non-exhaustive list of more specific examples of the computer readable storage medium includes the following: a portable computer diskette, a hard disk, a random-access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), a static random-access memory (SRAM), a portable compact disc read-only memory (CD-ROM), a digital versatile disk (DVD), a memory stick, and/or any suitable combination of the foregoing. A computer readable storage medium, as used herein, is not to be construed as being transitory signals per se, such as radio waves or other freely propagating electromagnetic waves, electromagnetic waves propagating through a waveguide or other transmission media (e.g., light pulses passing through a fiber-optic cable), or electrical signals transmitted through a wire. Rather, the computer readable storage medium is a non-transient (i.e., not-volatile) medium.

[00124] Computer readable program instructions described herein may be downloaded to respective computing/processing devices from a computer readable storage medium or to an external computer or external storage device via a network, for example, the Internet, a local area network, a wide area network and/or a wireless network. The network may comprise copper transmission cables, optical transmission fibers, wireless transmission, routers, firewalls, switches, gateway computers and/or edge servers. A network adapter card or network interface in each computing/processing device receives computer readable program instructions from the network and forwards the computer readable program instructions for storage in a computer readable storage medium within the respective computing/processing device.

[00125] Computer readable program instructions for carrying out operations of the present invention may be assembler instructions, instruction-set-architecture (ISA) instructions,

machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, or either source code or object code written in any combination of one or more programming languages, including an object oriented programming language such as Java, Smalltalk, C++ or the like, and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The computer readable program instructions may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider). In some embodiments, electronic circuitry including, for example, programmable logic circuitry, field-programmable gate arrays (FPGA), or programmable logic arrays (PLA) may execute the computer readable program instructions by utilizing state information of the computer readable program instructions to personalize the electronic circuitry, in order to perform aspects of the present invention.

[00126] Aspects of the present invention are described herein with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems), and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, may be implemented by computer readable program instructions.

[00127] These computer readable program instructions may be provided to a processor of a general-purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks. These computer readable program instructions may also be stored in a computer readable storage medium that may direct a computer, a programmable data processing apparatus, and/or other devices to function in a particular manner, such that the computer readable storage medium having instructions stored therein comprises an article of

manufacture including instructions which implement aspects of the function/act specified in the flowchart and/or block diagram block or blocks.

[00128] The computer readable program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other device to cause a series of operational steps to be performed on the computer, other programmable apparatus or other device to produce a computer implemented process, such that the instructions which execute on the computer, other programmable apparatus, or other device implement the functions/acts specified in the flowchart and/or block diagram block or blocks.

[00129] The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods, and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of instructions, which comprises one or more executable instructions for implementing the specified logical function(s). In some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, may be implemented by special purpose hardware-based systems that perform the specified functions or acts or carry out combinations of special purpose hardware and computer instructions.

[00130] The descriptions of the various embodiments of the present invention have been presented for purposes of illustration, but are not intended to be exhaustive or limited to the embodiments disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope and spirit of the described embodiments. The terminology used herein was chosen to best explain the principles of the embodiments, the practical application or technical improvement over technologies found in the marketplace, or to enable others of ordinary skill in the art to understand the embodiments disclosed herein.

CLAIMS

What is claimed is:

1. A method for secure comparison of confidential data, the method comprising using at least one hardware processor for:

computing a first set of encoded items by encoding each item of a first list of confidential data items;

computing a second set of encoded items by encoding of each item of a second list of confidential data items;

comparing each of element of the first set with each element of the second set; and indicating when a match is found,

wherein, during execution of the method:

a) a first computer performing the encoding of the first list is prevented access to the confidential data items of the second list, and

b) a second computer performing the encoding of the second list is prevented access to the confidential data items of the first list.

2. The method according to claim 1, wherein the computing of the first set and the computing of the second set is performed asynchronously.

3. The method according to claim 1, wherein the comparing, the computing of the first set, and the computing of the second set, are performed asynchronously.

4. The method according to claim 1, wherein the comparing is executed on a third computer, wherein the third computer is prevented access to the confidential digital data items from either the first list and the confidential digital data items of the second list.

5. The method according to claim 1, wherein the indicating is to at least one of: a first party and a second party, wherein the first party has access to the first list and is prevented from accessing the second list, and wherein the second party has access to the second list and is prevented from accessing the first list.

6. The method according to claim 1, wherein the encoding of the digital data items of at least one of the first and second lists is based on one function from the group consisting of a secure multi-party computation, a one-way function, a blind signature method, a Diffie-Hellman assumption, oblivious evaluation of a function, oblivious evaluation of a pseudo-random function, and a k-wise independent function.

7. The method according to claim 1, further comprising at least one of adding, deleting, and updating digital data items in the first or second encoded items lists.

8. The method according to claim 1, wherein the comparing of the encoded items is performed as a one-to-many comparison.

9. A computer program product for secure comparison of confidential data, the computer program product comprising a non-transitory computer-readable storage medium having program code embodied therewith, the program code executable by at least one hardware processor to perform the steps of:

computing a first set of encoded values by encoding of each item of a first list of confidential data values;

computing a second set of encoded values by encoding of each item of a second list of confidential data values;

comparing each of element of the first set with each element of the second set to produce matching elements; and

indicating the matching elements,

wherein, during execution of the method:

c) a first computer performing the encoding of the first list is prevented access to the confidential data value of the second list, and

d) a second computer performing the encoding of the second list is prevented access to the confidential data items of the first list.

10. The computer program product according to claim 9, wherein the computing of the first set and the computing of the second set is performed asynchronously.

11. The computer program product according to claim 9, wherein the comparing, the computing of the first set, and the computing of the second set are performed asynchronously.

12. The computer program product according to claim 9, wherein the comparing is executed on a third computer, wherein the third computer is prevented access to the confidential digital data items from either the first list and the confidential digital data items of the second list.

13. The computer program product according to claim 9, wherein the indicating is to at least one of a first party and a second party, wherein the first party has access to the first list and is prevented from accessing the second list, and wherein the second party has access to the second list and is prevented from accessing the first list.

14. The computer program product according to claim 9, wherein the encoding of the digital data items of at least one of the first and second lists is based one from the group consisting of a secure multi-party computation, a one-way function, a blind signature method, a Diffie-Hellman assumption, oblivious evaluation of a function, and a k-wise independent function.

15. The computer program product according to claim 9, further comprising at least one of adding, deleting, and updating digital data items in the first or second encoded items lists.

16. The computer program product according to claim 9, wherein the comparing of the encoded items is performed as a one-to-many comparison.

17. A system for secure comparison of confidential data, comprising:

a first computer configured to receive a first list of confidential digital data items; and

a second computer configured to receive a second list of confidential digital data items,

wherein:

the first and second computers are configured to communicate with each other over a computer network,

either (a) the first computer or (b) the first and second computer jointly are configured to compute an encoding of each item of the first list, to produce a first set of encoded items,

either (a) the second computer or (b) the first and second computer jointly are configured to compute an encoding of each item of the second list, to produce a second set of encoded items,

either (a) the first computer, (b) the second computer, or (c) a third computer is configured to compare the encoded items of the first list against the encoded items of the second list, and to indicate when a match is found, and

during the computation and comparison operations, the first and third computers are prevented access to any whole one of the confidential data items of the second list, and the second and third computers are prevented access to any whole one of the confidential data items of the first list.

18. The system according to claim 17, wherein at least one of:

(i) the list of the confidential digital data items of the first list are maintained in the first computer, and

(ii) the list of the confidential digital data items of the second list are maintained in the second computer.

19. The system according to claim 17, wherein the list of the secure encoding items of at least one of the first list and the second list are maintained in at least one of the first computer, the second computer, and the third computer.

20. The system according to claim 17, wherein at least one of the first computer and second computer automatically collects confidential items from other devices.

21. The system according to claim 17, wherein the computing of the encoded items of the first list and the computing of the encoded items of the second list is performed asynchronously.

22. The system according to claim 17, wherein the comparing, computing of the encoded items of the first list and the computing of the encoded items of the second list is performed asynchronously.

23. The system according to claim 17, wherein the comparing is executed on a third computer, wherein the third computer is prevented access to the confidential digital data items from either the first list and the confidential digital data items of the second list.

24. The system according to claim 17, wherein the indication about a match can be indicated just to the first computer, just to the second computer, or to both computers.

25. The system according to claim 17, wherein the encoding of the digital data items of at least one of the first and second lists is based on one function from the group consisting of a secure multi-party computation, a one-way function, a blind signature method, a Diffie-Hellman assumption, oblivious evaluation of a function, oblivious evaluation of a pseudo-random function, and a k-wise independent function.

26. The system according to claim 17, further comprising at least one of adding, deleting, and updating digital data items in the first or second encoded items lists.

27. The system according to claim 17, wherein the comparing of the encoded items is performed as a one-to-many comparison.

28. A computerized method for secure comparison of confidential digital data values, the computerized method comprising using at least one hardware processor for:

receiving, at each of a plurality of computers comprising a local confidential database, one set of a plurality of encoding processor instruction sets;

computing, by each corresponding computer, an encoding of at least one value of the corresponding local confidential database according to the corresponding set of encoding processor instructions, to produce encoded database values on each corresponding computer;

receiving at least some of the encoded database values;

comparing the received encoded database values to produce matching elements; and

issuing an indication of the matching elements.

29. The method according to claim 28, wherein the receiving further comprises receiving a set of encoded value comparison instructions, wherein the set of encoded value comparison instructions comprises a comparison rule, and wherein the comparing is performed using the set of encoded value comparison instructions.

30. The method according to claim 28, wherein the received encoded database values are compared with encoded database values derived from the local confidential database.

31. The method according to claim 28, wherein the receiving and comparing are performed on one of the plurality of computers.

32. The method according to claim 28, wherein the receiving and comparing are performed on a dedicated computer, wherein the dedicated computer is different from the plurality of computers.

33. A computer program product for secure comparison of confidential data, the computer program product comprising a non-transitory computer-readable storage medium having program code embodied therewith, the program code executable by at least one hardware processor to perform the steps of:

receiving, at each of a plurality of computers comprising a local confidential database, one set of a plurality of encoding processor instruction sets;

computing, by each corresponding computer, an encoding of at least one value of the corresponding local confidential database according to the corresponding set of encoding processor instructions, to produce encoded database values on each corresponding computer;

receiving at least some of the encoded database values;

comparing the received encoded database values to produce matching elements; and

issuing an indication of the matching elements.

34. The computer program product according to claim 33, wherein the receiving further comprises receiving a set of encoded value comparison instructions, wherein the set of encoded value comparison instructions comprises a comparison rule, and wherein the comparing is performed using the set of encoded value comparison instructions.

35. The computer program product according to claim 33, wherein the received encoded database values are compared with encoded database values derived from the local confidential database.

36. The computer program product according to claim 33, wherein the receiving and comparing are performed on one of the plurality of computers.

37. The computer program product according to claim 33, wherein the receiving and comparing are performed on a dedicated computer, wherein the dedicated computer is different from the plurality of computers.

38. A system for secure comparison of confidential data, comprising

a plurality of computers, each comprising (i) a at least one hardware processor, (ii) a non-transitory computer-readable storage medium comprising a local confidential database, and (iii) a network connection;

wherein each non-transitory computer-readable storage medium has program code embodied therewith, the program code executable by the respective at least one hardware processor to perform the steps of:

receiving, at each of the plurality of computers, one set of a plurality of encoding processor instruction sets;

computing, by each corresponding computer, an encoding of at least one value of the corresponding local confidential database according to the corresponding set of encoding processor instructions, to produce encoded database values on each corresponding computer;

receiving at least some of the encoded database values;

comparing the received encoded database values to produce matching elements; and

issuing an indication of the matching elements.

39. The system according to claim 38, wherein the receiving further comprises receiving a set of encoded value comparison instructions, wherein the set of encoded value comparison instructions comprises a comparison rule, and wherein the comparing is performed using the set of encoded value comparison instructions.

40. The system according to claim 38, wherein the received encoded database values are compared with encoded database values derived from the local confidential database.

41. The system according to claim 38, wherein the receiving and comparing are performed on one of the plurality of computers.

42. The system according to claim 38, wherein the receiving and comparing are performed on a dedicated computer, wherein the dedicated computer is different from the plurality of computers.

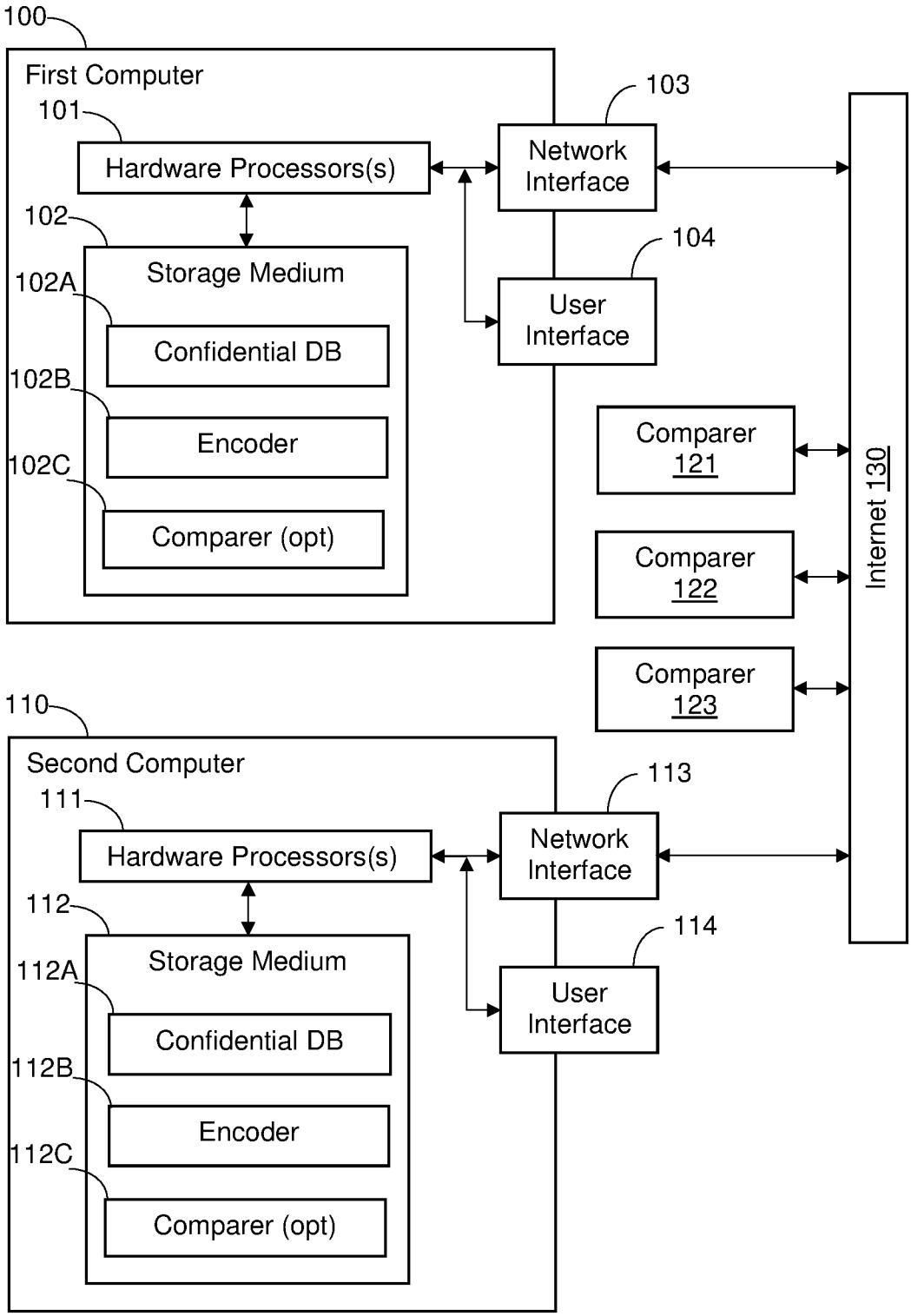
100
First Computer
101
Hardware Processors(s)
102
Storage Medium
102A
Confidential DB
102B
Encoder
102C
Comparer (opt)
103
Network Interface
104
User Interface
Comparer 121
Comparer 122
Comparer 123
Internet 130
110
Second Computer
111
Hardware Processors(s)
112
Storage Medium
112A
Confidential DB
112B
Encoder
112C
Comparer (opt)
113
Network Interface
114
User Interface

FIG. 1

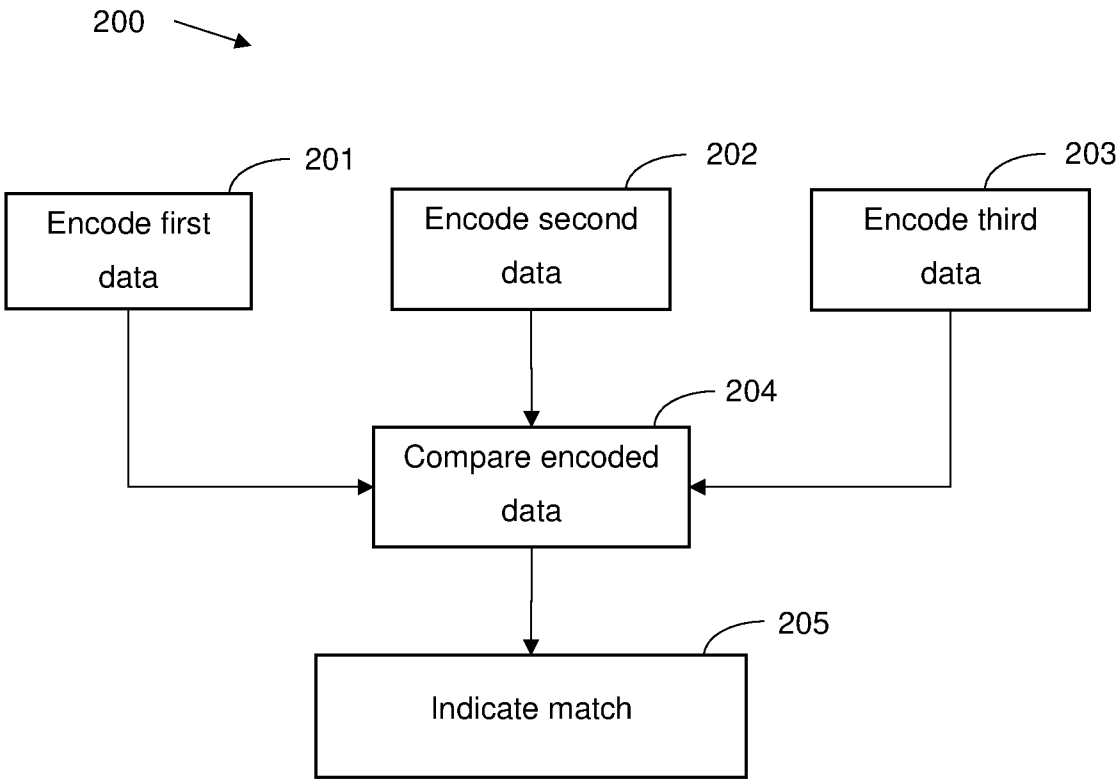
200
201
Encode first data
202
Encode second data
203
Encode third data
204
Compare encoded data
205
Indicate match

FIG. 2

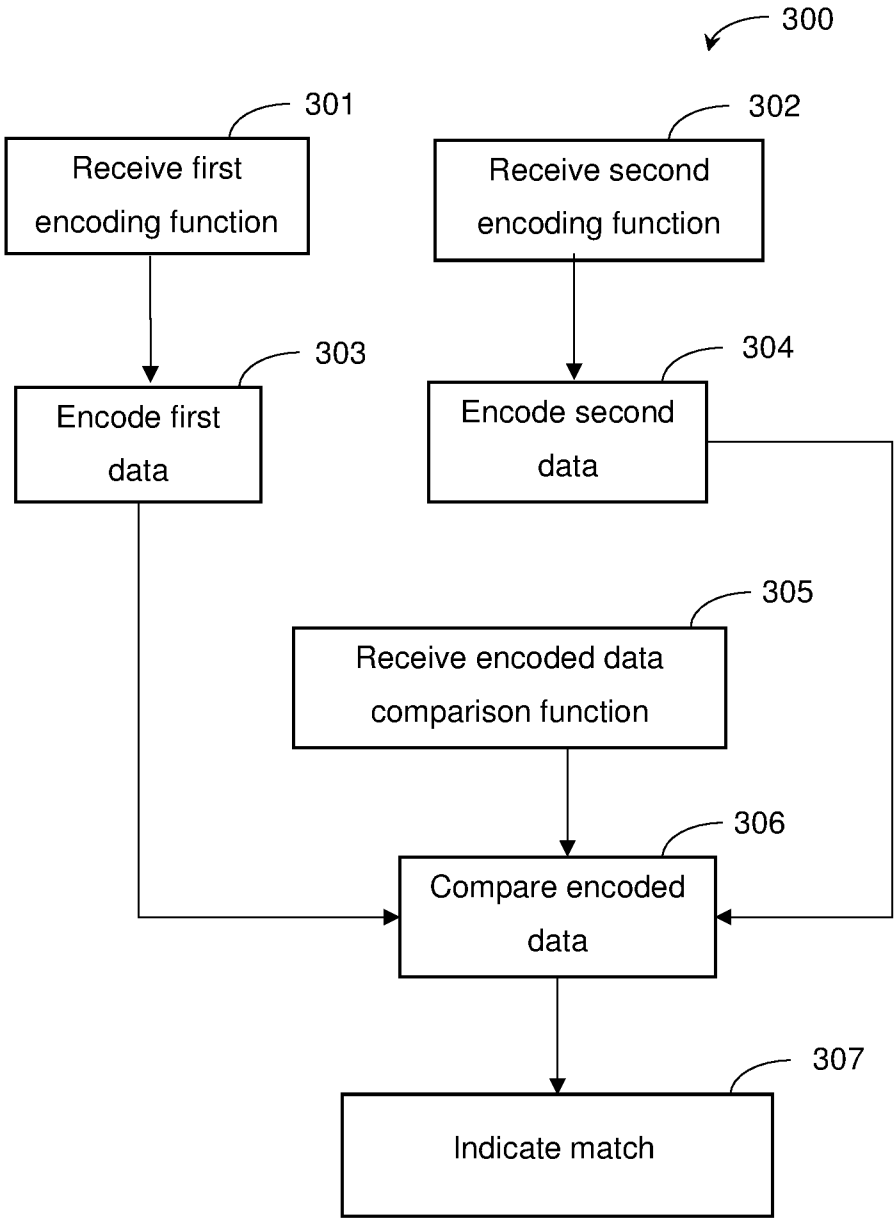
300
301
Receive first encoding function
302
Receive second encoding function
303
Encode first data
304
Encode second data
305
Receive encoded data comparison function
306
Compare encoded data
307
Indicate match

FIG. 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IL2017/050669

A. CLASSIFICATION OF SUBJECT MATTER
IPC (2017.01) G06F 21/60

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC (2017.01) G06F 21/60

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Databases consulted: Esp@cenet
Search terms used: part* match sensitive

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2016140348 A1 NAWAZ et al. 19 May 2016 (2016/05/19) (¶¶4,13,16,	1-42
A	US 2006245587 A1 Pinkas et al. 02 Nov 2006 (2006/11/02) abstract,¶¶13-14	1-42
A	WO 2015198098 A1 POURZANDI et al. 30 Dec 2015 (2015/12/30) The entire document	1-42
A	US 2011179274 A1 Veugen et al. 21 Jul 2011 (2011/07/21) The entire document	1-42

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
12 Sep 2017	17 Sep 2017

Name and mailing address of the ISA:	Authorized officer
Israel Patent Office Technology Park, Bldg.5, Malcha, Jerusalem, 9695101, Israel Facsimile No. 972-2-5651616	MAUDA Nissim Telephone No. 972-2-5651733

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/IL2017/050669

Patent document cited search report	Publication date	Patent family member(s)	Publication Date
US 2016140348 A1	19 May 2016	US 2016140348 A1	19 May 2016
		US 9443092 B2	13 Sep 2016
		EP 3024169 A1	25 May 2016
US 2006245587 A1	02 Nov 2006	US 2006245587 A1	02 Nov 2006
WO 2015198098 A1	30 Dec 2015	WO 2015198098 A1	30 Dec 2015
		EP 3161992 A1	03 May 2017
		US 2017124348 A1	04 May 2017
US 2011179274 A1	21 Jul 2011	US 2011179274 A1	21 Jul 2011
		US 8527765 B2	03 Sep 2013
		EP 2120393 A1	18 Nov 2009
		EP 2286540 A1	23 Feb 2011
		EP 2286540 B1	14 Sep 2016
		JP 2011521553 A	21 Jul 2011
		JP 5544355 B2	09 Jul 2014
		WO 2009139629 A1	19 Nov 2009