(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 3 506 547 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
03.07.2019 Bulletin 2019/27

(51) Int Cl.:
H04L 9/00 *(2006.01)* ***H04L 9/08*** *(2006.01)*

(21) Application number: **17210886.2**

(22) Date of filing: **28.12.2017**

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR
Designated Extension States:
BA ME
Designated Validation States:
MA MD TN

(71) Applicant: **Flytxt B.V.**
3439 MR Nieuwegein (NL)

(72) Inventor: **Veugen, Thijs**
2595 DA Den Haag (NL)

(74) Representative: **De Vries & Metman**
Overschiestraat 180
1062 XK Amsterdam (NL)

(54) PROVIDING SECURITY AGAINST USER COLLUSION IN DATA ANALYTICS USING RANDOM GROUP SELECTION

(57) Methods for secure random selection of t client devices which from a set of N client devices and methods for secure computation of inputs of t client devices which are randomly selected from a set of N client devices are described. Such random selection method may include determining an initial binary vector b of weight t by setting the first t bits to one: $b_i = 1$, $1 \leq i \leq t$, and all further bits to zero: $b_i = 0$, $t < i \leq N$; each client device i (i=1,...,N) of the set of N client devices jointly generating a random binary vector b of weight t in an obfuscated domain on the basis of the initial binary vector b including: determining a position n in the binary vector; determining a random number r in {n,n+1,...N}; and, using the random number to swap binary values at positions n and r of the binary vector b.

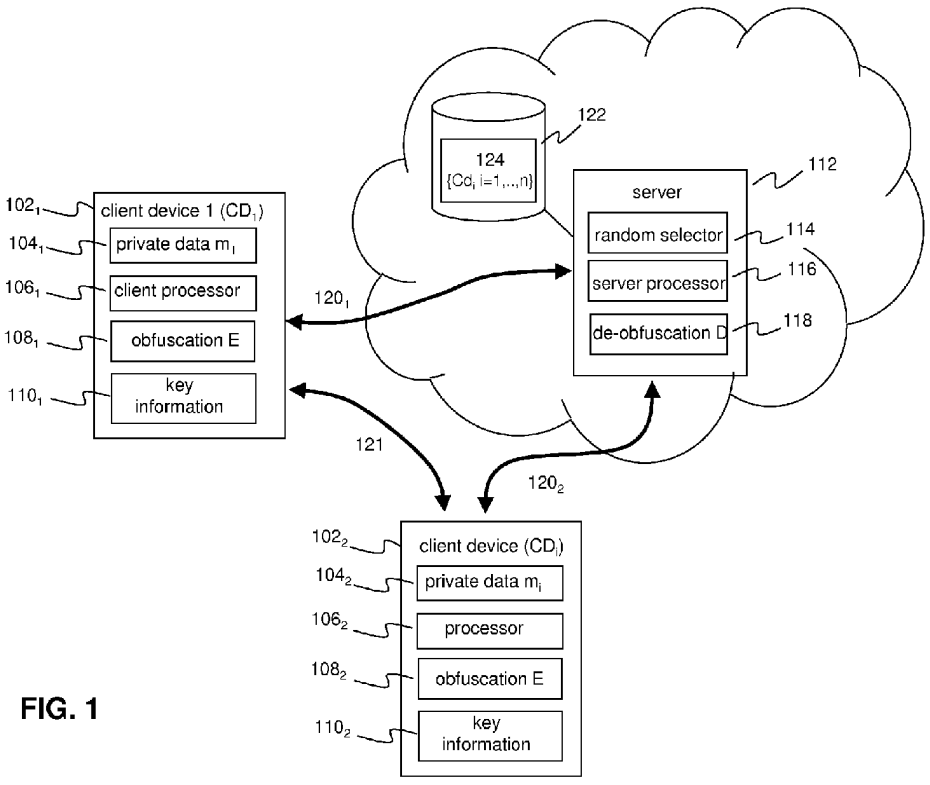

FIG. 1

EP 3 506 547 A1

Description

Field of the invention

[0001] The invention relates to providing security against user collusion in data analytics using random group selection, and, in particular, though not exclusively, to methods and systems for random group selection, methods and system for secure computation of inputs using random group selection, client devices and server devices adapted to execute such methods and a computer program product adapted to execute such methods.

Background of the invention

[0002] Currently the number of data processing schemes that require the processing of large sets of privacy-sensitive data is growing exponentially. Commercial, governmental and academic organizations make decisions or verify or disprove models, theories or hypotheses using big data analytics. These techniques generally include collecting, organizing and analysing large sets of privacy-sensitive data in order to discover descriptive, predictive or even prescriptive insights about a group or groups of people, organisations or companies. This information may be of substantial commercial value. For example, a recommender system may include one or more network servers that run an algorithm to generate personal suggestions for individual users on the basis of data of a large group of users. The data is collected by client applications running on (mobile) devices of the users and transmitted to the server using a suitable client-server protocol.

[0003] Data processing schemes used in big data analytics are based on multiple sets of privacy-sensitive user inputs, wherein each set of user-inputs may be provided by a selected group of users, wherein the data processing may include computations, e.g. one or more arithmetical operations such as addition, multiplication, comparison, etc., of the collected data. Privacy-preserving protocols are used in order to eliminate or at least limit information leakage during the processing and transmission of the data. Typically, these techniques include obfuscating the privacy-sensitive data prior to the data processing and communication, which is typically performed in an untrustworthy environment.

[0004] WO2013066176 and WO2013066177 describe a privacy-preserving protocol for use in a recommender system wherein inputs from selected groups of users may be aggregated and processed in the encrypted domain using a cryptographic privacy-preserving protocol which may include the use of a homomorphic cryptographic system. The problem of such approach however is that despite the use of cryptographic techniques, information leakage is still possible. For example, Kononchuk et al. "Privacy-preserving user data oriented services for groups with dynamic participation", in Computer Security (ESORICS) Lecture Notes in Computer Science, 8134, Springer, Berlin, 2013) have shown that repeating such a protocol several times with groups of colluding users may cause information to leak away.

[0005] Kononchuk et al. suggested the so-called random user selection approach in order to cope with the problem of user collusion leading to information leakage. In this approach, a random subset of users is generated for the computation of an output function (e.g. computing suggestions in a recommendation system) on the basis of a random bit vector, wherein each element of the vector indicates whether a particular user is selected or not for computing a current group service. This selection vector is kept hidden from all parties, including the server that is computing the aggregation. The use of the random user selection approach is further described in the article by Veugen et al., "Improved privacy of dynamic group services", EURASIP journal on information security, March 2017.

[0006] In Kononchuk et al. the random bits generation is solved by having each user randomly generating a permutation of length N, concatenating these N permutations to one joint random permutation, and using this to permute an initial vector containing exactly t ones. One problem associated with this solution is that when the number of permutations N becomes large (e.g. a random selection from a set of a million users or more) the permutation process becomes communicational and computational complex. Random bit generation by means of permutation matrices requires N matrix multiplications, wherein each matrix multiplication requires exchange of information between the participating client devices. In practical implementations however client devices are heterogeneous devices, typically mobile devices having limited resources. Hence, the communication and computational complexity of the protocol should be minimized without compromising the security of the protocol.

[0007] Hence, from the above follows a need in the art for improved methods and systems for privacy-preserving random group selection. In particular, there is a need in the art for methods and systems that enable secure random selection of a group of client devices and secure aggregation of privacy sensitive data of a randomly selected group of client devices. This efficiency is typically important for data analytics, as more efficient algorithms allow for recommender systems to be more responsive to changes in user behavior; and packaging of more information, such as user behavior history, leads to more accurate predictions.

Summary of the invention

[0008] As will be appreciated by one skilled in the art, aspects of the present invention may be embodied as a system, method or computer program product. Accordingly, aspects of the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, mi-

cro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a "circuit", "module" or "system." Functions described in this disclosure may be implemented as an algorithm executed by a microprocessor of a computer. Furthermore, aspects of the present invention may take the form of a computer program product embodied in one or more computer readable medium(s) having computer readable program code embodied, e.g., stored, thereon.

[0009] Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples (a non- exhaustive list) of the computer readable storage medium would include the following: an electrical connection having one or more wires, 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), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination of the foregoing. In the context of this document, a computer readable storage medium may be any tangible medium that can contain, or store a program for use by, or in connection with, an instruction execution system, apparatus, or device.

[0010] A computer readable signal medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electromagnetic, optical, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport a program for use by, or in connection with, an instruction execution system, apparatus, or device.

[0011] Program code embodied on a computer readable medium may be transmitted using any appropriate medium, including, but not limited to, wireless, wireline, optical fiber, cable, RF, etc., or any suitable combination of the foregoing. Computer program code for carrying out operations for aspects of the present invention may be written in any combination of one or more programming languages, including an object oriented programming language such as Java(TM), Smalltalk, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The program code 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).

[0012] Aspects of the present invention are described below 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, can be implemented by computer program instructions. These computer program instructions may be provided to a processor, in particular a microprocessor or central processing unit (CPU), 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, other programmable data processing apparatus, or other devices create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0013] These computer program instructions may also be stored in a computer readable medium that can direct a computer, other programmable data processing apparatus, or other devices to function in a particular manner, such that the instructions stored in the computer readable medium produce an article of manufacture including instructions which implement the function/act specified in the flowchart and/or block diagram block or blocks.

[0014] The computer program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other devices to cause a series of operational steps to be performed on the computer, other programmable apparatus or other devices to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0015] 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 code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that, in some alternative implementations, the functions noted in the blocks 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 illustrations, and combinations of blocks in the block diagrams and/or flowchart illustrations, can be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

[0016] It is an objective of the invention to reduce or eliminate at least one of the drawbacks known in the prior art. In an aspect, the invention may relate to a method for secure computation of inputs of t client devices which are randomly selected from a set of N client devices, wherein the method may comprise: each client device i (i=1,...,N) of the set of N client devices generating a random binary vector b of weight t in an obfuscated domain, preferably an encrypted domain or a secret-sharing domain, the generating including using random numbers to randomly swap bit values bi at different positions i (i=1,...,N) in the binary vector; each client device i (i=1,...,N) transforming an input value x_i into the obfuscated domain and determining the product $b_i \cdot x_i$ of bit value bi at position i of the random binary vector and the input value x_i in the obfuscated domain; and, performing a secure computation in the obfuscated domain on the basis of the products $b_i \cdot x_i$ (i=1,...,N).

[0017] Hence, the method provides a computationally and communicationally efficient scheme for secure selection of a random group of client devices from a set of N client devices. Additionally, the method provides a computationally and communicationally efficient scheme for performing secure computations on the basis of the input values of the randomly selected client devices. Both the random selection and the computations are executed in the obfuscated domain so that neither the client devices, nor devices external to the client devices, e.g. a server, have knowledge about the group that is selected for the computation, or about the computation itself (e.g. the input values and/or the outcome of the computation). The secure random bit generation protocol takes not more than 3tN secure multiplications. This is a more efficient way (i.e. compared to Kononchuk et al) of securing the privacy of users is secured against collusion by malicious users.

[0018] The invention is particular useful for big data analytics applications that involve multi-party computations in which the privacy of users needs to be secured. These applications include (but not limited to) online or cloud-based processing of private-sensitive data for recommender systems, processing metrics in television and multimedia applications (e.g. audience ratings of channel usage in television broadcast or streaming applications), processing of financial data of commercial and/or institutional organisations, etc.

[0019] In an embodiment, generating a random binary vector may include: determining an initial binary vector b of weight t by setting the first t bits to one: $b_i = 1$, $1 \leq i \leq t$, and all further bits to zero: bi = 0, $t < i \leq N$; generating a random binary vector on the basis of the initial binary vector, the generating including determining a position n in the binary vector and determining a random number r in {n,n+1,...N} and using the random number to swap binary values at positions n and r of the binary vector b.

[0020] In an embodiment, the swapping of the bits may be based on a delta function δ_i^n, wherein $\delta_i^n = 1$, if a random value $r_n = i$, and $\delta_i^n = 0$, for all other i positions in the vector.

[0021] In an embodiment, the delta function δ_i^n may be defined on the basis of a polynomial function $D_i^n(x)$:

$$D_i^n(x) = \prod_{j=n, j\neq i}^{N} \frac{x-j}{i-j} = \sum_{j=0}^{N-n} d_j x^j$$

wherein d_j represent coefficients of the Lagrange polynomial $D_i^n(x)$.

[0022] In an embodiment, the obfuscated domain may be based on a homomorphic cryptosystem, preferably an additively homomorphic cryptosystem.

[0023] In an embodiment, the random generated binary vector may include encrypted random bits $[b_i]$, $1 \leq i \leq N$, such that $\sum_i b_i = t$; wherein the client device i processes its input x_i on the basis of the binary vector of encrypted bits by computing $[x_i \cdot b_i] = [b_i]^{x_i}$; and, wherein the secure computation in the obfuscated domain is based on the computed values $[x_i \cdot b_i]$ (i=1,...,N).

[0024] In an embodiment, the method may further include: each of the client devices i transmitting the computed value $[x_i \cdot b_i]$ to a server, preferably an aggregation server; and, the server performing the secure computation on the basis of the computed values $[x_i \cdot b_i]$ (i=1,...,N).

[0025] In an embodiment, the method may further include: each of the client devices transmitting the computed value $[x_i \cdot b_i]$ to one or more client devices; and, the one or more client devices performing the secure computation on the basis of $[x_i \cdot b_i]$ (i=1,...,N).

[0026] In an embodiment, the obfuscated domain may be based on a secret-sharing system, preferably the secret-sharing system being based on a modulo computation using a fixed prime p.

[0027] In an embodiment, the random generated binary vector may include secret-shared random bits $\langle b_i \rangle$, $1 \leq i \leq N$, such that $\sum_i b_i = t$; wherein the client device i determines a secret share $\langle x_i \rangle$ using the fixed prime p, and the clients jointly compute $\langle b_i \cdot x_i \rangle$; and wherein the secure computation in the obfuscated domain is based on the computed values $\langle b_i \cdot x_i \rangle$ (i=1,...,N).

[0028] In a further aspect, the invention may relate to a system for secure computation of inputs of t client devices which are randomly selected from a set of N client devices. In an embodiment, the system may comprise: a set of client devices i (i=1,...,N), each client device comprising a computer readable storage medium having computer readable program code embodied therewith, the program code including an obfuscation function, preferably a homomorphic encryption function, or a secret-

sharing function, for performing computations in an obfuscated domain and a processor, preferably a microprocessor, coupled to the computer readable storage medium, wherein responsive to executing the first computer readable program code, wherein the processor is configured to perform executable operations comprising: generating a random binary vector b of weight t in an obfuscated domain, preferably an encrypted domain or a secret-sharing domain, the generating including using random numbers to randomly swap bit values b_i at different positions i (i=1,...,N) in the binary vector; transforming an input value x_i into the obfuscated domain, and determining the product $b_i \cdot x_i$ of bit value bi at position i of the random binary vector and the input value x_i in the obfuscated domain; and, transmitting the computed product $b_i \cdot x_i$ to a server system or to a further client device, wherein the server or the further client device is configured to perform a secure computation in the obfuscated domain on the basis of the products $b_i \cdot x_i$ (i=1,...,N) computed by each client device i (i=1,...,N).

[0029] In a further aspect, the invention may relate to client apparatus for use in a system for secure computation of inputs of t client devices, which are randomly selected from a set of N client devices. In an embodiment, the client device may comprise: a computer readable storage medium having computer readable program code embodied therewith, the program code including an obfuscation function, preferably a homomorphic encryption function or a secret-sharing function, for performing computations in an obfuscated domain and a processor, preferably a microprocessor, coupled to the computer readable storage medium, wherein responsive to executing the first computer readable program code, wherein the processor is configured to perform executable operations comprising: generating a random binary vector b of weight t in the obfuscated domain, the generating including using random numbers to randomly swap bit values bi at different positions i (i=1,...,N) in the binary vector; and, transforming an input value x_i into the obfuscated domain and determining the product $b_i \cdot x_i$ of bit value bi at position i of the random binary vector and the input value x_i in the obfuscated domain.

[0030] In an embodiment, the executable operations may further comprise: transmitting the computed product $b_i \cdot x_i$ to a server system or to a further client device, wherein the server or the further client device is configured to perform a secure computation in the obfuscated domain on the basis of the products $b_i \cdot x_i$ (i=1,...,N) computed by each client device i (i=1,...,N).

[0031] In a further aspect, the invention relates to a method for secure random selection of t client devices from a set of N client devices. In an embodiment, the method may comprise: determining an initial binary vector b of weight t by setting the first t bits to one: $b_i = 1$, $1 \leq i \leq t$, and all further bits to zero: $b_i = 0$, $t < i \leq N$; each client device i (i=1,...,N) of the set of N client devices jointly generating a random binary vector b of weight t in an obfuscated domain on the basis of the initial binary vector b, a bit i in the random binary vector b signalling if client device i is selected or not, the generation of the random binary vector including: determining a position n in the binary vector; determining a random number r in {n,n+1,...N}; and, using the random number to swap binary values at positions n and r of the binary vector b.

[0032] The secure random selection process may be used in any multi-party computation scheme that requires secure selection of t users (client devices) from a group of N users (client devices).

[0033] In an embodiment, the swapping of the binary values at positions n and r may be based on a delta function δ_i^n, wherein $\delta_i^n = 1$, if a random value $r_n = i$, and $\delta_i^n = 0$, for all other i positions in the vector. In an embodiment, the delta function δ_i^n may be defined on the basis of a polynomial function $D_i^n(x)$:

$$D_i^n(x) = \prod_{j=n, j\neq i}^{N} \frac{x-j}{i-j} = \sum_{j=0}^{N-n} d_j x^j$$

wherein d_j represent coefficients of the Lagrange polynomial $D_i^n(x)$.

[0034] In an embodiment, the obfuscated domain may be based on a homomorphic cryptosystem, preferably an additive homomorphic cryptosystem.

[0035] In an embodiment, determining a random number r may further include: each client device i generating 1 random bits, $0 \leq j < 1$, where 1 is such that $2^{l-1} \leq N - n < 2^l$, encrypting the random bits α_i^j using the homomorphic cryptosystem; each client device i computing l encrypted random bits $[\alpha^j]$, $0 \leq j < 1$, wherein $[\alpha^j]=[\alpha_1^j \oplus \alpha_2^j \oplus ... \oplus \alpha_N^j])$; each client device i computing an encrypted random number [r] on the basis of the encrypted random bits $[\alpha^j]$.

[0036] In an embodiment, the computation of the encrypted random number may include

$$[r] = \left[\sum_{j=0}^{l-1} \alpha^j \cdot 2^j\right] = \prod_{j=0}^{l-1} \left[\alpha_j\right]^{2^j}.$$

[0037] In an embodiment, the obfuscated domain may be based on a secret-sharing system, preferably the secret-sharing system being based on a modulo computation using a fixed prime p.

[0038] In an embodiment, the determining of a random number r may further include: each client device i may generating 1 random bits α_i^j , $0 \leq j < 1$, where 1 is such that $2^{l-1} \leq N - n < 2^l$, transforming the random bits to the secretly-shared domain; the client devices jointly compute 1 secretly-shared random bits $\langle \alpha^j \rangle$, $0 \leq j < 1$ wherein $\langle \alpha^j \rangle = \langle \alpha_1^j \oplus \alpha_2^j \oplus ... \oplus \alpha_N^j \rangle$;

each client device i computing the secretly shared random number $\langle r \rangle$ on the basis of the secretly-shared random bits $\langle \alpha^j \rangle$. In an embodiment, the computation of the secretly-shared random number may include

$$\langle r \rangle = \sum_{j=0}^{l-1} \langle \alpha_j \rangle \cdot 2^j .$$

[0039] The invention may also relate to a program product comprising software code portions configured for, when run in the memory of a computer, executing any of the method steps described above.

[0040] The invention will be further illustrated with reference to the attached drawings, which schematically will show embodiments according to the invention. It will be understood that the invention is not in any way restricted to these specific embodiments.

Brief description of the drawings

[0041]

Fig. 1 schematically depicts a system for secure computation of inputs of a randomly selected group of client devices according to an embodiment of the invention;

Fig. 2 schematically depicts randomly selected group of client devices in an obfuscated domain according to an embodiment of the invention;

Fig. 3 depicts a method for secure computation of inputs of a randomly selected group of client devices according to an embodiment of the invention;

Fig. 4 depicts a method for secure computation of inputs of a randomly selected group of client devices according to another embodiment of the invention;

Fig. 5 depicts a method for secure computation of inputs of a randomly selected group of client devices according to yet another embodiment of the invention;

Fig. 6 is a block diagram illustrating an exemplary data computing system that may be used for executing methods and software products described in this disclosure.

Detailed description

[0042] The embodiments in this disclosure generally relate privacy-preserving random group selection schemes and multi-party computation schemes, such as used in data-analytics, using such privacy-preserving random group selection, in particular privacy-preserving multi-party computation schemes wherein a random securely selected group of client devices may jointly compute a function on the basis of privacy-sensitive user data. **Fig. 1** schematically depicts a system for secure computation of inputs of a randomly selected group of client devices according to an embodiment of the invention. The system may comprise client devices $\mathbf{102}_{1,2}$ and, in some embodiments, at least one server **112**, e.g. an aggregation server. Each client device may comprise a processor $\mathbf{106}_{1,2}$ adapted to run one or more applications, e.g. software applications including software code when executed by the processor configuring the client device to collect or generate privacy sensitive user data, e.g. medical data, financial data, user behaviour data, commercially sensitive data, etc. Hereafter, privacy sensitive user data associated with client device i may be referred to in short as private data m_i.

[0043] A client device may be implemented on a user apparatus, e.g. a smart phone, a computer, a server, a (smart) television, a vehicle or any consumer or industrial electronic apparatus that is configured to process and/or generate private data. Client devices may typically comprise a wireless or a hardwired communication interface allowing client devices to communicate with each other or with network elements, e.g. servers, in the network. The client devices and the server may use a secure protocol, in particular a secure multi-party computation protocol based on an obfuscation scheme, e.g. an encryption scheme or secret sharing scheme, which ensures secure communication between the client devices and the aggregation server **112**. The server may comprise a processor **116** adapted to process the private data in the obfuscated domain. Communication between the client devices and the server device in the network may be based on a suitable client server protocol $\mathbf{120}_{1,2}$ e.g. an HTTP protocol or the like. Further, during the execution of the secure multi-party computation protocols described in this application, client devices may exchange information with each other using a central model, e.g. a trusted server or using a decentralized communication model on the basis of a suitable peer-to-peer protocol **121**, e.g. the well-known BitTorrent protocol or a derivative thereof.

[0044] In an embodiment, when the system of **Fig. 1** is configured to process data in the encrypted domain, the secure protocol may be based on a homomorphic cryptosystem. The homomorphic cryptosystem may comprise an encryption function E and a decryption function D, wherein a homomorphically encrypted value E(x) may be notated in short as [x], wherein the homomorphic properties of the cryptosystem allow certain computations in the encrypted domain. For example, in an embodiment, the homomorphic cryptosystem may be an additively homomorphic crypto system having the property $[x_{1]} \cdot [x_2] = [x_1 + x_2]$. The well-known ElGamal cryptosystem may be designed to have additively homomorphic properties. The processing of the data by the aggregation server in the encrypted domain may include one or more computations, e.g. additions and/or multiplications. In an embodiment, the computations may be part of a data processing algorithm that is used by a data analytics application.

[0045] In an embodiment, the homomorphic cryptosystem may be configured as a so-called homomorphic threshold cryptosystem. Examples of homomorphic threshold cryptosystems are described in the article by Y. Desmedt and Y. Frankel, with title "threshold cryptosystems", CRYPTO 1989, P.308-315, Springer-Verlag. The threshold homomorphic cryptosystem is associated with key information $\mathbf{110}_{1,2}$, including public key information, e.g. a public encryption key e that is shared by the

client devices and the aggregator server, and secret key information including a secret decryption key d, which may be secret-shared among the client devices. Each client device i may be provided with a decryption share d_i of the secret decryption key d, wherein the decryption operation needs to be jointly performed by sufficiently many clients.

[0046] An example of a privacy-preserving computing system that is based on a threshold homomorphic cryptosystem is described in a related European patent application EP 17177653.7 with title "Privacy preserving computation protocol for data analytics", which is herewith incorporated by reference into the description of this application.

[0047] In another embodiment, when the system of **Fig. 1** is configured to process data in the secret-shared domain, the secure protocol may be based on a secret-sharing system. An example of such secret-sharing system is described in the article by Damgard et al., "Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation", In: Halevi, S., Rabin, T. (eds.) TCC 2006, LNCS, vol. 3876, pp. 285-304. Springer, Heidelberg. In such secret-sharing scheme, values are usually computed on the basis of a modulo operation using a fixed prime p. Having a secret sharing $\langle x \rangle$ means that each client device i has a (meaningless) share x_i, such that $\sum_i x_i \bmod p = x$. Adding two secret-shared values is simply locally adding the shares. Multiplying two secret-shared values $\langle x \rangle$ and $\langle y \rangle$ however, requires a protocol and communication between the parties. As a prerequisite, the parties should hold secret-sharings of three random numbers a, b, and c, such that $c = (a \cdot b) \bmod p$.

[0048] In the system of **Fig. 1**, the client devices may form a group of N client devices, wherein each client device CD_i of index i (i=1,...,n) may comprise an obfuscation function $\mathbf{108_{1,2}}$, e.g. encryption function E of an homomorphic cryptosystem, in order to obfuscate private data m_i $\mathbf{104_{1,2}}$ before sending the data to the aggregation server. Further, the aggregator server may comprise a de-obfuscation function **118**, e.g. a decryption function D of the homomorphic cryptosystem, so that - after processing the obfuscated private data in the obfuscated domain - it can access the cleartext result of the processing by transforming the obfuscated result back to the cleartext domain. A storage medium **122**, e.g. a database, connected to the aggregation server may comprise client information **124** regarding client devices that may participate in the aggregation process. Such client information may include location information, e.g. an URL or a network address, and client identifiers $\{CD_i\}$ for uniquely identifying each client device. Alternatively, and/or additionally, a client device (identified by a client identifier) may be associated with an index i, preferably a countable index i, (i.e. a quantity which can take on a set of integer values and is used to designate one out of a number of possible values associated with this value). This index may hereafter be referred to as a client index. The client information may be used by the server to establish data connections with the different client devices.

[0049] The secure protocol is configured as a privacy-preserving protocol allowing a server processor **116** to securely process, e.g. aggregate, private data m_i of a group of (at least) t client devices randomly selected from N client devices without the risk that the server and/or client devices learn the individual private data m_i and without leaking information of user inputs to sources. Selection of the client devices by the aggregation server may be realized by a selector **114**.

[0050] In many big data applications, the aggregation process is repeated multiple times wherein each time the selector **114** may select a different group of client devices from the N client devices. However, as shown by Kononchuk et al., when the computing system repeats the aggregation process a number of times for different groups of users, information about the private data may leak away. For example, when the aggregator server would collude with t - 1 users, it is possible to obtain the private input of any other user in this way.

[0051] This information leakage problem may be addressed by implementing the selector **114** as a secure random group selection processor. This selector may trigger the N client devices to execute a random group selection protocol. As shown in **Fig. 2**, this protocol is adapted to randomly select a subgroup H **204** of t client devices from a group G **202** of N client devices, such that neither the user of the aggregator server **206**, nor the users of the client devices can influence the result. In an embodiment, such random selection may be realized by the client devices jointly generating a set of (homomorphically) encrypted bits [bi], $1 \leq i \leq N$, such that $\sum_i b_i = t$. To avoid the 'dummy user' attack, the protocol for generating the bits should be secure, even when only one user is honest.

[0052] Prior art solutions include a random group selection process that involves the execution of permutations in the encrypted domain resulting in a computation-heavy protocol, which renders the protocol not, or at least less, suitable for practical applications. It is an aim of the embodiments in this application to provide an efficient privacy-preserving random group selection process which eliminates, or at least reduces, the risk of leakage of privacy sensitive information in multi-party computation schemes, which are often used in data analytics applications. In an embodiment, such random group selection process may be used in a privacy-preserving computing system as described with reference to **Fig. 1**. Such system may be part of a multi-party computation system, e.g. a recommender system as e.g. described with reference to WO2013066176 and WO2013066177.

[0053] The random group selection processes used in the embodiment of this application may be based on a set of bits, e.g. a binary vector bi, $1 \leq i \leq N$, which is used to identify which client device in a set of N client devices is selected. For example, setting the k-th bit of the vector to one (i.e. $b_k = 1$) may indicate that client device with

client index i=k has been selected to participate in a multi-party computation scheme, e.g. secure data aggregation process. Hence, randomly selecting a group of client devices includes securely and randomly setting t bits in the vector to one. The random group selection process according to the invention thus includes the generation of a binary vector of weight t, using a random bit generation method in a domain in which the bits are obfuscated, e.g. the encrypted domain or the secret-sharing domain.

[0054] The random binary vector generation may include a swapping process, wherein random numbers are used to swap two bits in the binary vector. This process may include an initial step in which a binary vector of weight t is initialized by setting the first t bits to one: $b_i = 1$, $1 \leq i \leq t$, and all further bits to zero: $b_i = 0$, $t < i \leq N$. Thereafter, a bit swapping process may be repeated t times. An example of such bit swapping protocol may look as follows.

[0055] For n = 1 to t do:

1. Client devices jointly generate a random number r in {n,n+1, ...N};
2. Each client device swaps the binary values at positions n and r of vector b.

[0056] After execution of this bit swapping protocol, each bit bi is one with probability t/N. It only requires the generation of t random numbers and performing t swaps instead of generating N random permutations and N times permuting all items. The randomly generated vector represents a randomly selected group of client devices, wherein a bit at the k-th position in the vector signal indicates whether client device with index k is part of the selected group. In order to avoid leakage of the selection process to unauthorized users, the swapping protocol is executed in an obfuscated domain. Transforming the swapping protocol to such obfuscated domain, e.g. the encrypted domain or the secret-sharing domain, is however not a trivial exercise. This will be illustrated in more detail in the embodiments hereunder.

[0057] In an embodiment, the bit swapping protocol may be executed in the encrypted domain using e.g. a homomorphic encryption system. In that case, a binary vector b of encrypted bits may be generated in which the first t encrypted bits may be set to one: $[b_i] = [1]$, $1 \leq i \leq t$ and the rest of the encrypted bits may be set to zero: $[b_i] = [0]$, $t < i \leq N$. Further, t encrypted random numbers $[r_n]$, $1 \leq n \leq t$, need to be generated by the system such that r_n is uniformly drawn from the set {n, n + 1, ... N}.

[0058] Thereafter, a bit swapping process may be executed in which the encrypted random numbers $[r_n]$ are used to swap encrypted bits in the binary vector.

[0059] In order to process bits of the binary vector, a suitable bit manipulation function may be used. In an embodiment, a delta function δ_i^n may be used for processing bits in the binary vector, wherein $\delta_i^n = 1$, if $r_n = i$, and 0, for all other i positions in the vector. Hence, the delta function sets the bit at position r_n in the binary vector to one, and to 0 for all other positions.

[0060] In order to transform the delta function to an obfuscated domain, the function may be defined on the basis of a Lagrange interpolation scheme wherein a function $D_i^n(x)$ may define the following polynomial:

$$D_i^n(x) = \prod_{j=n, j\neq i}^{N} \frac{x-j}{i-j} = \sum_{j=0}^{N-n} d_j x^j$$

Here, the coefficients d_j represent coefficients of the Lagrange polynomial $D_i^n(x)$. Based on this function, δ_i^n can be constructed as follows: $\delta_i^n = D_i^n(r_n)$, $n \leq i \leq N$, which represents a bit on position r_n in the binary vector. This function can be transformed into the encrypted domain using an additively homomorphic cryptosystem and the Lagrange coefficients d_j:

$$[\delta_i^n] = \prod_{j=0}^{N-n} [r^j]^{d_j}$$

[0061] As shown by this expression, the homomorphic encryption transforms the summation into a product. This encrypted delta function may be used to process, e.g. swap, bits in the encrypted domain. This way, a bit swapping process may be executed in the encrypted domain, which includes swapping t encrypted bits of the binary vector on the basis of encrypted random numbers.

[0062] A process of securely generating random bits, e.g. in the form of a random binary vector, according to an embodiment of the invention may look as follows:

For n = 1 to t do

1. a set of N client devices jointly generate an encrypted random number [r], such that $r \in \{n,n+1,N\}$;
2. the client devices jointly compute $[r^j]$, for each j, $1 \leq j \leq N - n$, using secure multiplications;

a. for i= n to N do:

i. each user computes the coefficients d_j, $0 \leq j \leq N - n$, of the polynomial $D_i^n(x)$;

ii. each client device computes $[\delta_i^n] = \prod_{j=0}^{N-n} [r^j]^{d_j}$

b. each client device computes $[b] = \prod_{i=n}^{N} [b_i \cdot \delta_i^n]$, after N - n + 1 joint secure multiplications, such that $b = b_r$;

c. set $[b_n] = [b]$ ($b_n \leftarrow b_r$);

d. for i = n to N do

each client device computes

$[b_i] = [\delta_i^n] \cdot [b_i] \cdot [\delta_i^n \cdot b_i]^{-1}$ after a joint secure multiplication of $[\delta_i^n]$ and $[b_i]$ ($b_r \leftarrow b_n$).

[0063] In step 2.d of the protocol above, the new value of bi becomes $\delta_i^n + b_i - \delta_i^n \cdot b_i$, which equals 1, if $\delta_i^n = 1$, and bi, if $\delta_i^n = 0$. Since $b_n = 1$, and $\delta_i^n = 1$ exactly when i=r, we get $b_r \leftarrow b_n$, where the other bi remain unaltered. This secure random bit generation protocol takes not more than 3tN secure multiplications and t random number generations. In contrast, the generation by means of permutation matrices as described in the prior art requires N matrix multiplications, which sums up to N^4 secure multiplications.

[0064] In the above-described random bit generation protocol, the N client devices need to jointly generate an encrypted random number r, $r \in \{n,n+1,...,N\}$.

[0065] In an embodiment, such encrypted random number may be generated by the client devices as follows:

1. each client device i may generate 1 random bits α_i^j, $0 \leq j < 1$, where 1 is such that $2^{l-1} \leq N - n < 2^l$, and encrypts the random bits using an homomorphic cryptosystem;
2. the client devices jointly compute 1 random bits $[\alpha^j]$, $0 \leq j < 1$, which can be computed using N secure multiplications, since the exclusive-or $\alpha_1 \oplus \alpha_2 = \alpha_1 + \alpha_2 - 2\alpha_1 \cdot \alpha_2$ (here α^j are truly random as long as the user of one client device is honest: $[\alpha^j]=[a_1^j \oplus \alpha_2^j \oplus ... \oplus \alpha_N^j]$);
3. the client devices jointly compute $[r] = \left[\sum_{j=0}^{l-1} \alpha^j \cdot 2^j\right] = \prod_{j=0}^{l-1} [\alpha_j]^{2^j}$;
4. the client devices check whether $r \leq N - n$, by securely computing the encrypted comparison bit $[\gamma]$ where $\gamma = 1$, if $r \leq N - n$, and 0, otherwise (this step may be executed within l secure multiplications, since the encrypted bits $[_\alpha^j]$ of r are given);
5. the client devices decrypt γ and if γ=1, $[r] \leftarrow [r+n] = [r] \cdot [n]$.

[0066] If $\gamma = 1$, this protocol securely generates an encrypted random number r from the set {n,n+1,...,N}. The computation of the comparison is known per se. An example of such computation is described in B. Schoenmakers and P. Tuyls, Practical Two-Party Computation based on the Conditional Gate, ASIACRYPT 2004, Lecture Notes in Computer Science 3329 (2004) 119136. Springer-Verlag. Typically, to generate many bounded random numbers, per random number four executions of the protocol above is sufficient.

[0067] In the above-described protocol, the client devices use a secure multiplication protocol, which is described hereunder in greater detail. In this scheme, initially, the client devices hold encryptions of additively homomorphically encrypted numbers [x] and [y], and would like to securely compute an encryption $[x \cdot y]$ of the product. To that end, the client devices may execute the following steps:

1. each client device i may generate two large random numbers r_i^x and r_i^y, encrypt both numbers $[r_i^x]$ and $[r_i^y]$, and broadcast the two encryptions to all other client devices;
2. each client device may compute two encrypted random numbers $[r^x] = [\sum_i r_i^x] = \prod_i [r_i^x]$ and $[r^y] = [\sum_i r_i^y] = \prod_i [r_i^y]$;
3. the client devices may then compute $[x+r^x] = [x] \cdot [r^x]$ and $[y+r^y] = [y] \cdot [r^y]$, and decrypt both values into $x + r^x$ and $y + r^y$;
4. client device i may further compute $[z_i] = ([r^y] \cdot [y])^{r_i^x} \cdot [x]^{r_i^y}$, and broadcast it to all other client devices;
5. the client devices then compute $[z] = [(x + r^x) \cdot (y + r^y)] \cdot (\Pi_i [z_i])^{-1}$.

[0068] In the scheme above, the integers $x + r^x$ and $y + r^y$ may be safely decrypted, since the values of x and y are additively blinded by a large random number, which is unknown to each party. It can be shown that $z = x \cdot y$: $(x+r^x) \cdot (y+r^y) = xy + xr^y + r^x y + r^x r^y = xy + \sum_i z_i$.

[0069] In an embodiment, the above-described privacy-preserving random group selection process may be used for secure aggregation of private data. **Fig. 3** depicts a method for secure computation of inputs of a randomly selected group of client devices according to an embodiment of the invention. In a first step **300**, a server may initiate a random group selection process in order to securely aggregate data from the selected group of client devices. In step **302**, the client devices may jointly generate encrypted random bits b_i wherein b_i, $1 \leq i \leq N$, forms a random binary vector of obfuscated bits, the random binary vector being generated by randomly swapping bits in the obfuscated domain. Thereafter, each client device may generate an obfuscated user input x_i and determine the product $x_i \cdot b_i$ in the obfuscated domain (step **304**). Each client device then transmits the obfuscated product $x_i \cdot b_i$ to an aggregation server (step **306**), which may use the products to perform secure computation in the obfuscated domain (step 308). The obfuscated result y of secure computation may be transmitted back to (part of) the client devices, which may de-obfuscate the obfuscated result y (step **308**).

[0070] The method allows secure computation of user inputs of t client devices, which are randomly selected from a group of N client device. The process includes a secure selection of client devices in the obfuscated domain, so that neither the client devices, nor the server, know which inputs are used in the computation. Here, the number t may be selected on the basis of various considerations including e.g. the computation and communication time that is needed in order to process data of the selected client devices, the trust in the network, etc. The selection includes an efficient bit swapping process. The process may be implemented using different obfuscation schemes.

[0071] **Fig. 4** depicts a method for secure computation of inputs of a randomly selected group of client devices according to another embodiment of the invention. In this embodiment, a homomorphic cryptosystem is used to obfuscate the privacy-sensitive user data and the processing of these user data. As shown in the figure, in a first step **400**, a server may initiate a random group selection process in order to securely aggregate data from a selected group of client devices. Thereafter, the following steps may be executed:

1.N client devices may jointly generate encrypted random bits $[b_i]$, $1 \leq i \leq N$, such that $\sum_i b_i = t$ (step **402**);
2. client device i may process its input x_i on the basis of the binary vector of encrypted bits by computing $[x_i \cdot b_i] = [b_i]^{x_i}$ (step **404**), and transmit the computed value $[x_i \cdot b_i]$ to a server, e.g. an aggregation server (step **406**);
3. the server may process the inputs: $[y] = [\sum_i x_i \cdot b_i] = \Pi_i [x_i \cdot b_i]$ (step **408**), and transmit the result to one or more client devices (step **410**).

[0072] In case threshold decryption is used, the aggregator may request t users to decrypt [y]. In an embodiment, this may be combined with verifiable aggregation to assure that the aggregator added all inputs. Such scheme is described in a related European patent application EP 17177653.7 with title "Privacy preserving computation protocol for data analytics". This way, the aggregator obtains $y = \sum_{i \in H} x_i$, where the set $H = \{i | b_i = 1\}$ is randomly chosen, and hidden from all parties.

[0073] In a further embodiment, a secret-sharing based secure multi-party computation system may be used to generate the random bits. The random bits bi may be computed and are secret-shared between N client devices. If desired, it is possible to transform such a shared secret to a homomorphically encrypted bit $[b_i]$. This way, an aggregation server may add up encrypted inputs (as described hereunder in more detail).

[0074] When using secret-sharing, (as e.g. described in the article by Damgard et al., "Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation"), each secret value x is secret-shared which is indicated by the notation $\langle x \rangle$.

[0075] Similar to the above-described embodiment related to homomorphic encryption, the main problem is how to securely generate N secret-shared bits $\langle b_i \rangle$, such that $\sum_i b_i = t$. Also in this case, prior art solutions suggesting to generate N random permutations of length N, and concatenating these to one random permutation, will lead to very inefficient and computational heavy processes. Translating the random bit generation process of the invention into the secret-sharing domain will provide a very efficient and secure random group selection process.

[0076] The protocol for securely multiplying two secret-shared numbers however differs from a secure multiplication based on (threshold) homomorphic encryption. In particular, in a secret-sharing scheme, values are computed on the basis of a modulo of a fixed prime p. Hence, having a secret share $\langle x \rangle$ means that each party i has a (meaningless) share x_i, such that $\sum_i x_i \bmod p = x$. Adding two secret-shared values is simply a client device locally adding the shares. Multiplying two secret-shared values $\langle x \rangle$ and $\langle y \rangle$ however, requires a protocol and communication between client devices. As a prerequisite, the client devices should hold secret-sharings of three random numbers a, b, and c, such that $c = (a \cdot b) \bmod p$. Then, a multiplication may look as follows:

1. the client devices locally compute $\langle d \rangle = \langle x \rangle + \langle a \rangle$, and $\langle e \rangle = \langle y \rangle + \langle b \rangle$;
2.the client devices reveal the secrets d and e by broadcasting their shares of these two numbers;
3. the client devices compute $\langle z \rangle = de - d \cdot \langle b \rangle - e \cdot \langle a \rangle + \langle c \rangle$.

wherein $z = x \cdot y$, since $x \cdot y = (d-a)(e-b) = de - db - ae + ab$.

[0077] Given the secure multiplication protocol for secret-sharings, the secure random bit generation protocol in the encrypted domain can be easily translated into the secret-share domain by using $\langle . \rangle$ instead of [.].

[0078] The secure random bit generation protocol in the secret-share domain may include an initialization process wherein a binary vector of secret-shared bits may be generated, in which the first t secret-shared bits may be set to one: $\langle b_i \rangle = \langle 1 \rangle$, $1 \leq i \leq t$ and the rest of the secretly shared bits may be set to zero: $\langle b_i \rangle = \langle 0 \rangle$, $t < i \leq N$.

[0079] Thereafter, a bit swapping process may be executed in the secret-shared domain, which includes swapping t secret-shared bits of the binary vector in a similar way as described above with reference to the swapping process in the encrypted domain. In an embodiment, the process may look as follows:
For n = 1 to t do

1. a set of N client devices jointly generate a secret-shared random number $\langle r \rangle$, such that $r \in \{n, n+1, ..., N\}$;
2. the client devices jointly compute $\langle r^j \rangle$, for each j, $1 \leq j \leq N - n$, using secure multiplications;

a. for i= n to N do:

i. each client device computes the coefficients d_j, $0 \leq j \leq N - n$, of the polynomial $D_i^n(x)$;
ii. each client device computes $\langle \delta_i^n \rangle = \sum_{j=0}^{N-n} \langle r^j \rangle \cdot d_j$;

b. each client device computes $\langle b \rangle = \sum_{i=n..N} \langle \delta_i^n \cdot b_i \rangle$, after N - n + 1 joint secure multiplications, such that $b = b_r$;
c. set $\langle b_n \rangle = \langle b \rangle$;
d. for i = n to N do
each client device computes $\langle b_i \rangle = \langle \delta_i^n \rangle + \langle b_i \rangle - \langle \delta_i^n$

$\cdot b_i\rangle$ after a joint secure multiplication of $\langle \delta_i^n \rangle$ and $\langle b_i \rangle$.

[0080] In the above-described random bit generation protocol, the client device need to jointly generate an encrypted random number r, $r \in \{n, n+1, ..., N\}$. In an embodiment, such encrypted random number may be generated by the client devices as follows:

1. each client device i may generate 1 random bits α_i^j, $0 \leq j < 1$, where 1 is such that $2^{l-1} \leq N - n < 2^l$, and transform the random bits to the secretly-shared domain;
2. the client devices jointly compute 1 random bits $\langle \alpha^j \rangle$, $0 \leq j < l$: $\langle \alpha^j \rangle = \langle \alpha_1^j \oplus \alpha_2^j \oplus ... \oplus \alpha_N^j \rangle$, which can be computed within N secure multiplications, since the exclusive-or $\alpha_1 \oplus \alpha_2 = \alpha_1 + \alpha_2 - 2\alpha_1 . \alpha_2$ (here α^j are truly random as long as the user of one client device is honest);
3. the client devices jointly compute $\langle r \rangle = \sum_{j=0}^{l-1} \langle \alpha_j \rangle \cdot 2^j$;
4. the client devices check whether $r \leq N - n$, by securely computing the comparison bit $\langle \gamma \rangle$, where $\gamma = 1$, if $r \leq N - n$, and 0, otherwise. This can be done within 1 secure multiplications, since the secret-shared bits $\langle \alpha^j \rangle$ of r are given;
5. the client devices de-obfuscates γ and if $\gamma = 1$, $\langle r \rangle \leftarrow \langle r+n \rangle = \langle r \rangle + \langle n \rangle$.

[0081] If $\gamma = 1$, this protocol securely generates a secret-shared random number r from the set $\{n, n+1, ..., N\}$. Typically, to generate many bounded random numbers, per random number four executions of the protocol above should suffice.

In general, an addition of two encrypted values [x] and [y] is computed by $[x+y] = [x] \cdot [y]$, whereas secret-shared values are just (locally) added: $(x+y) = \langle x \rangle + \langle y \rangle$. Similarly, $[c \cdot x] = [x]^C$ becomes $\langle c \cdot x \rangle = \langle x \rangle \cdot c$, which is locally multiplying (modulo p) each share with the known number c.

[0082] **Fig. 5** depicts a method for secure computation of inputs of a randomly selected group of client devices according to yet another embodiment of the invention. In this embodiment, a secret-sharing scheme is used to obfuscate the privacy-sensitive user data and the processing of these user data. In this embodiment, instead of a server, an aggregation server, calculating the aggregated user input, the aggregation may be implemented in a decentralized scheme wherein the aggregation is performed jointly by the client devices. In that case, either a client device or an external device, e.g. a server may trigger the start of the random group selection (step **500**). Thereafter, the following steps may be executed:

1. N client devices may jointly generate encrypted random bits $\langle b_i \rangle$, $1 \leq i \leq N$, such that $\sum_i b_i = t$ (step **502**) ;
2. client device i may determine a secret-sharing $\langle x_i \rangle$ using the fixed prime p and private share x_i, jointly with the other client devices (in the semi-honest model, which is easily achieved by setting the local shares of the other players to zero), and jointly compute $\langle b_i \cdot x_i \rangle$ by means of a secure multiplication (step **504**);
3. the secret-shared result $\langle y \rangle$ may be determined on the basis of the shares, e.g. by locally adding the corresponding shares: $y = \sum_i x_i b_i$ (step **506**).

Such decentralized scheme may also be implemented in the encrypted domain wherein the aggregation calculation $[y] = [\sum_i x_i \cdot b_i] = \Pi_i [x_i \cdot b_i]$ may be performed by one or more client devices, or all client devices of the N client devices. In that case, each client device is configured to transmit or broadcast the securely computed products $[b_i \cdot x_i]$ to the other N-1 client devices in the set. Hence, in case an external device (or one of the client devices) needs the result of the aggregation, the one or more client devices can perform the aggregation calculation, and decrypt the aggregation result.

[0083] An advantage of such scheme is that it is no longer necessary to verify the aggregation step performed by the aggregator server. Hence, this way, computations on the secret (and randomised) user data, which is usually done by a central server, may be performed by the client devices. This enables schemes performing all kinds of computations on the user data, without restricting to the homomorphic property of the encryption system.

[0084] The systems and methods for privacy-preserving computation of private data as described in this application are particularly useful for implementation in data analytics applications, which require the processing of large amounts of privacy sensitive data. An example of such data analytics application may be a secure privacy-preserving recommender system, wherein a group of participants (voters) may register with the platform, and install an electronic voting application on their computer or mobile device.

[0085] The problem of preserving data privacy of users, who repeatedly join the execution of a service with different groups, is present in many data analytics applications requiring multi-party computation, including for example recommendation systems (see Z Erkin, et al, Generating private recommendations efficiently using homomorphic encryption and data packing, IEEE Trans. Inform. Forensics Secur. 7(3), 1053-1066, 2012), aggregating data in smart grids, reputation systems (see e.g. J Kacprzyk, Group decision making with a fuzzy linguistic majority, Fuzzy Sets Syst, 18, 105-118, 1986), unsupervised machine learning (see e.g. F Anselmi et al., Unsupervised learning of invariant representations with low sample complexity: the magic of sensory cortex or a new framework for machine learning? J. Theor. Comput. Sci. 633(C), 112-121 (2016). CBMM, MIT, arXiv:1311.4158v5), collective decision-making (see e.g.A Mathes, Folksonomies-cooperative classification and

communication through shared metadata, Comput. Mediated Commun. 47, 1-28, 2004). nr. 10), data clustering (see e.g. Z Erkin et al., in IEEE International Workshop on Information Forensics and Security, Privacy-preserving user clustering in a social network, IEEE, Washington, 2009), and social classification (see e.g. H Kargupta et al, in ICDM, IEEE Computer Society, On the privacy preserving properties of random data perturbation techniques, IEEE, Washington, 2003, pp. 99-106). The embodiments in this disclosure can be advantageously used in such data analytics applications to eliminate or at least reduce undesired information leakage and to secure the privacy of users against collusion by malicious users during the execution of such services.

[0086] The systems and methods described in this application may be used in any data analytics application that requires an aggregated result (e.g. a sum and/or product) of private data of individual users. For example, possible data analytics applications may include the processing of medical data, e.g. online processing of clinical data or medical records of patients, processing data of voting and secret ballot schemes, processing metrics in television and multimedia applications (e.g. audience ratings of channel usage in television broadcast or streaming applications), processing of financial data of commercial and/or institutional organisations, etc.

[0087] **Fig. 6** is a block diagram illustrating exemplary data processing systems described in this disclosure. Data processing system **600** may include at least one processor **602** coupled to memory elements **604** through a system bus **606**. As such, the data processing system may store program code within memory elements **604**. Further, processor **602** may execute the program code accessed from memory elements **604** via system bus **606**. In one aspect, data processing system may be implemented as a computer that is suitable for storing and/or executing program code. It should be appreciated, however, that data processing system **600** may be implemented in the form of any system, including a processor and memory, that is capable of performing the functions described within this specification.

[0088] Memory elements **604** may include one or more physical memory devices such as, for example, local memory **608** and one or more bulk storage devices **610**. Local memory may refer to random access memory, or other non-persistent memory device(s) generally used during actual execution of the program code. A bulk storage device may be implemented as a hard drive or other persistent data storage device. The processing system **600** may also include one or more cache memories (not shown) that provide temporary storage of at least some program code in order to reduce the number of times program code must be retrieved from bulk storage device **610** during execution.

[0089] Input/output (I/O) devices depicted as input device **612** and output device **614** optionally can be coupled to the data processing system. Examples of input device may include, but are not limited to, for example, a keyboard, a pointing device such as a mouse, or the like. Examples of output device may include, but are not limited to, for example, a monitor or display, speakers, or the like. Input device and/or output device may be coupled to data processing system either directly or through intervening I/O controllers. A network adapter **616** may also be coupled to data processing system to enable it to become coupled to other systems, computer systems, remote network devices, and/or remote storage devices through intervening private or public networks. The network adapter may comprise a data receiver for receiving data that is transmitted by said systems, devices and/or networks to said data and a data transmitter for transmitting data to said systems, devices and/or networks. Modems, cable modems, and Ethernet cards are examples of different types of network adapter that may be used with data processing system **650**.

[0090] As pictured in **FIG. 6**, memory elements **604** may store an application **618**. It should be appreciated that data processing system **600** may further execute an operating system (not shown) that can facilitate execution of the application. Application, being implemented in the form of executable program code, can be executed by data processing system **600**, e.g., by processor **602**. Responsive to executing application, data processing system may be configured to perform one or more operations to be described herein in further detail.

[0091] In one aspect, for example, data processing system **600** may represent a client data processing system. In that case, application **618** may represent a client application that, when executed, configures data processing system **600** to perform the various functions described herein with reference to a "client". Examples of a client can include, but are not limited to, a personal computer, a portable computer, a mobile phone, or the like.

[0092] In another aspect, data processing system may represent a server. For example, data processing system may represent an (HTTP) server in which case application **618**, when executed, may configure data processing system to perform (HTTP) server operations. In another aspect, data processing system may represent a module, unit or function as referred to in this specification.

[0093] The terminology used herein is for the purpose of describing particular embodiments only, and is not intended to be limiting of the invention. As used herein, the singular forms "a", "an", and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms "comprises" and/or "comprising", when used in this specification, specify the presence of stated features, integers, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components, and/or groups thereof.

[0094] The corresponding structures, materials, acts, and equivalents of all means or step plus function elements in the claims below are intended to include any

structure, material, or act for performing the function in combination with other claimed elements as specifically claimed. The description of the present invention has been presented for purposes of illustration and description, but is not intended to be exhaustive or limited to the invention in the form 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 invention. The embodiment was chosen and described in order to best explain the principles of the invention and the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.

Claims

1. A method for secure computation of inputs of t client devices which are randomly selected from a set of N client devices comprising:

   each client device i (i=1,...,N) of the set of N client devices jointly generating a random binary vector b of weight t in an obfuscated domain, preferably an encrypted domain or a secret-sharing domain, wherein bit i in the random binary vector b signals if client device i is selected or not and wherein, during the generation of the random binary vector, random numbers are used to randomly swap bit values b_i at different positions i (i=1,...,N) in the binary vector;
   each client device i (i=1,...,N) transforming an input value x_i into the obfuscated domain and determining the product $b_i \cdot x_i$ of bit value bi at position i of the random binary vector and the input value x_i in the obfuscated domain;
   performing a secure computation in the obfuscated domain on the basis of the products $b_i \cdot x_i$ (i=1,...,N).

2. A method according to claim 1 wherein generating a random binary vector b of weight t in an obfuscated domain includes:

   determining an initial binary vector b of weight t by setting the first t bits to one: $b_i = 1$, $1 \le i \le t$, and all further bits to zero: $b_i = 0$, $t < i \le N$;
   generating a random binary vector on the basis of the initial binary vector, the generating including determining a position n in the binary vector and determining a random number r in {n,n+1,...N} and using the random number to swap binary values at positions n and r of the binary vector b.

3. Method according to claims 1 or 2 wherein the swapping of the bits is based on a delta function δ_i^n, wherein $\delta_i^n = 1$, if a random value $r_n = i$, and $\delta_i^n = 0$, for all other i positions in the vector, preferably the delta function δ_i^n being defined on the basis of a polynomial function $D_i^n(x)$:

$$D_i^n(x) = \prod_{j=n, j\neq i}^{N} \frac{x-j}{i-j} = \sum_{j=0}^{N-n} d_j x^j$$

   wherein d_j represent coefficients of the Lagrange polynomial $D_i^n(x)$.

4. Method according to any of claims 1-3 wherein the obfuscated domain is based on a homomorphic cryptosystem, preferably an additive homomorphic cryptosystem.

5. Method according to claim 4 wherein the random generated binary vector includes encrypted random bits $[b_i]$, $1 \le i \le N$, such that $\sum_i b_i = t$; wherein the client device i processes its input x_i on the basis of the binary vector of encrypted bits by computing $[x_i \cdot b_i] = [b_i]^{x_i}$; and, wherein the secure computation in the obfuscated domain is based on the computed values $[x_i \cdot b_i]$ (i=1,...,N).

6. The method according to claims 3 or 4 further including:

   each of the client devices i transmitting the computed value $[x_i \cdot b_i]$ to a server, preferably an aggregation server; and,
   the server performing the secure computation on the basis of the computed values $[x_i \cdot b_i]$ (i=1,...,N).

7. The method according to claims 3 or 4 further including:

   each of the client devices transmitting the computed value $[x_i \cdot b_i]$ to one or more client devices; and,
   the one or more client devices performing the secure computation on the basis of $[x_i \cdot b_i]$ (i=1,...,N).

8. Method according to any of claims 1-3 wherein the obfuscated domain is based on a secret-sharing system, preferably the secret-sharing system being based on a modulo computation using a fixed prime p.

9. Method according to claim 8 wherein the random generated binary vector includes secretly-shared random bits $\langle b_i \rangle$, $1 \le i \le N$, such that $\sum_i b_i = t$; wherein the client device i determines a secret sharing $\langle x_i \rangle$ using the fixed prime p, and uses $\langle x_i \rangle$ to compute $\langle x_i$

$\cdot b_i\rangle$; and wherein the secure computation in the obfuscated domain is based on the computed values $\langle b_i \cdot x_i\rangle$ (i=1,...,N).

10. A system for secure computation of inputs of t client devices which are randomly selected from a set of N client devices comprising:
a set of client devices i (i=1,...,N), each client device comprising a computer readable storage medium having computer readable program code embodied therewith, the program code including an obfuscation function, preferably a homomorphic encryption function or a secret-sharing function, for performing computations in an obfuscated domain and a processor, preferably a microprocessor, coupled to the computer readable storage medium, wherein responsive to executing the first computer readable program code, wherein the processor is configured to perform executable operations comprising:

generating a random binary vector b of weight t in an obfuscated domain, preferably an encrypted domain or a secret-sharing domain, the generating including using random numbers to randomly swap bit values bi at different positions i (i=1,...,N) in the binary vector;
transforming an input value x_i into the obfuscated domain, and determining the product $b_i \cdot x_i$ of bit value bi at position i of the random binary vector, and the input value x_i in the obfuscated domain; and,
transmitting the computed product $b_i \cdot x_i$ to a server system or to a further client device, wherein the server or the further client device is configured to perform a secure computation in the obfuscated domain on the basis of the products $b_i \cdot x_i$ (i=1,...,N) computed by each client device i (i=1,...,N).

11. A client apparatus for use in a system for secure computation of inputs of t client devices which are randomly selected from a set of N client devices, preferably a system according to claim 10, comprising:
a computer readable storage medium having computer readable program code embodied therewith, the program code including an obfuscation function, preferably a homomorphic encryption function or a secret-sharing function, for performing computations in an obfuscated domain and a processor, preferably a microprocessor, coupled to the computer readable storage medium, wherein responsive to executing the first computer readable program code, wherein the processor is configured to perform executable operations comprising:

generating a random binary vector b of weight t in the obfuscated domain, the generating including using random numbers to randomly swap bit values bi at different positions i (i=1,...,N) in the binary vector;
transforming an input value x_i into the obfuscated domain and determining the product $b_i \cdot x_i$ of bit value bi at position i of the random binary vector and the input value x_i in the obfuscated domain; and, optionally,
transmitting the computed product $b_i \cdot x_i$ to a server system or to a further client device, wherein the server or the further client device is configured to perform a secure computation in the obfuscated domain on the basis of the products $b_i \cdot x_i$ (i=1,...,N) computed by each client device i (i=1,...,N).

12. A method for secure random selection of t client devices from a set of N client devices comprising:

determining an initial binary vector b of weight t by setting the first t bits to one: bi = 1, $1 \leq i \leq t$, and all further bits to zero: bi = 0, $t < i \leq N$;
each client device i (i=1,...,N) of the set of N client devices jointly generating a random binary vector b of weight t in an obfuscated domain on the basis of the initial binary vector b, a bit i in the random binary vector b signalling if client device i is selected or not, the generation of the random binary vector including:

determining a position n in the binary vector;
determining a random number r in {n,n+1,...N}; and,
using the random number to swap binary values at positions n and r of the binary vector b.

13. Method according to claim 12 wherein the swapping of the binary values at positions n and r is based on a delta function δ_i^n, wherein $\delta_i^n = 1$, if a random value $r_n = i$, and $\delta_i^n = 0$, for all other i positions in the vector, preferably the delta function δ_i^n being defined on the basis of a polynomial function $D_i^n(x)$:

$$D_i^n(x) = \prod_{j=n, j\neq i}^{N} \frac{x-j}{i-j} = \sum_{j=0}^{N-n} d_j x^j$$

wherein d_j represent coefficients of the Lagrange polynomial $D_i^n(x)$.

14. Method according to claims 12 or 13 wherein the obfuscated domain is based on a homomorphic cryptosystem, preferably an additive homomorphic cryptosystem and wherein determining a random number r further includes:

each client device i generating 1 random bits , $0 \leq j < 1$, where 1 is such that $2^{l-1} \leq N - n < 2^l$,
encrypting the random bits α_i^j using the homomorphic cryptosystem;
each client device i computing 1 encrypted random bits $[\alpha^j]$, $0 \leq j < 1$, wherein $[\alpha^j] = [\alpha_1^j \oplus \alpha_2^j \oplus ... \oplus \alpha_N^j])$;
each client device i computing an encrypted random number [r] on the basis of the encrypted random bits $[\alpha^j]$, preferably the computing including

$$[r] = \left[\sum_{j=0}^{l-1} \alpha^j \cdot 2^j\right] = \prod_{j=0}^{l-1} [\alpha_j]^{2^j}.$$

15. Method according to claims 12 or 13 wherein the obfuscated domain is based on a secret-sharing system, preferably the secret-sharing system being based on a modulo computation using a fixed prime p; and wherein determining a random number r further includes:

each client device i may generating 1 random bits α_i^j, $0 \leq j < 1$, where 1 is such that $2^{l-1} \leq N - n < 2^l$,
transforming the random bits to the secretly-shared domain;
the client devices jointly compute 1 secretly-shared random bits $\langle \alpha^j \rangle$, $0 \leq j < 1$ wherein $\langle \alpha^j \rangle = \langle \alpha_1^j \oplus \alpha_2^j \oplus ... \oplus \alpha_N^j \rangle$;
each client device i computing the secretly shared random number $\langle r \rangle$ on the basis of the secretly shared random bits $\langle \alpha^j \rangle$, preferably the computing including $\langle r \rangle = \sum_{j=0}^{l-1} \langle \alpha_j \rangle \cdot 2^j$.

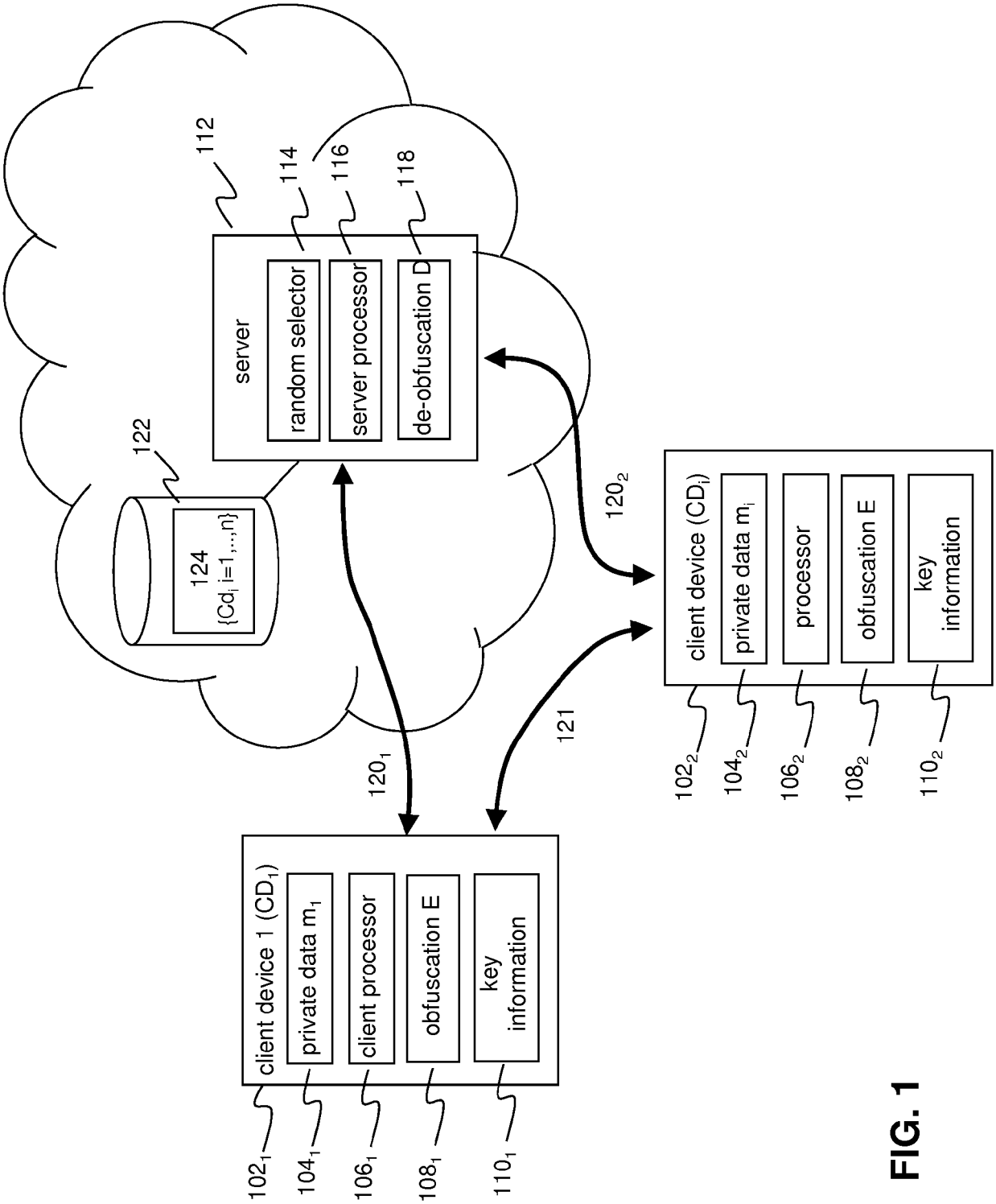
112
114
116
118
server
random selector
server processor
de-obfuscation D
122
124
{Cd_i i=1,..,n}
120_2
client device (CD_i)
private data m_i
processor
obfuscation E
key information
102_2
104_2
106_2
108_2
110_2
121
120_1
client device 1 (CD_1)
private data m_1
client processor
obfuscation E
key information
102_1
104_1
106_1
108_1
110_1

FIG. 1

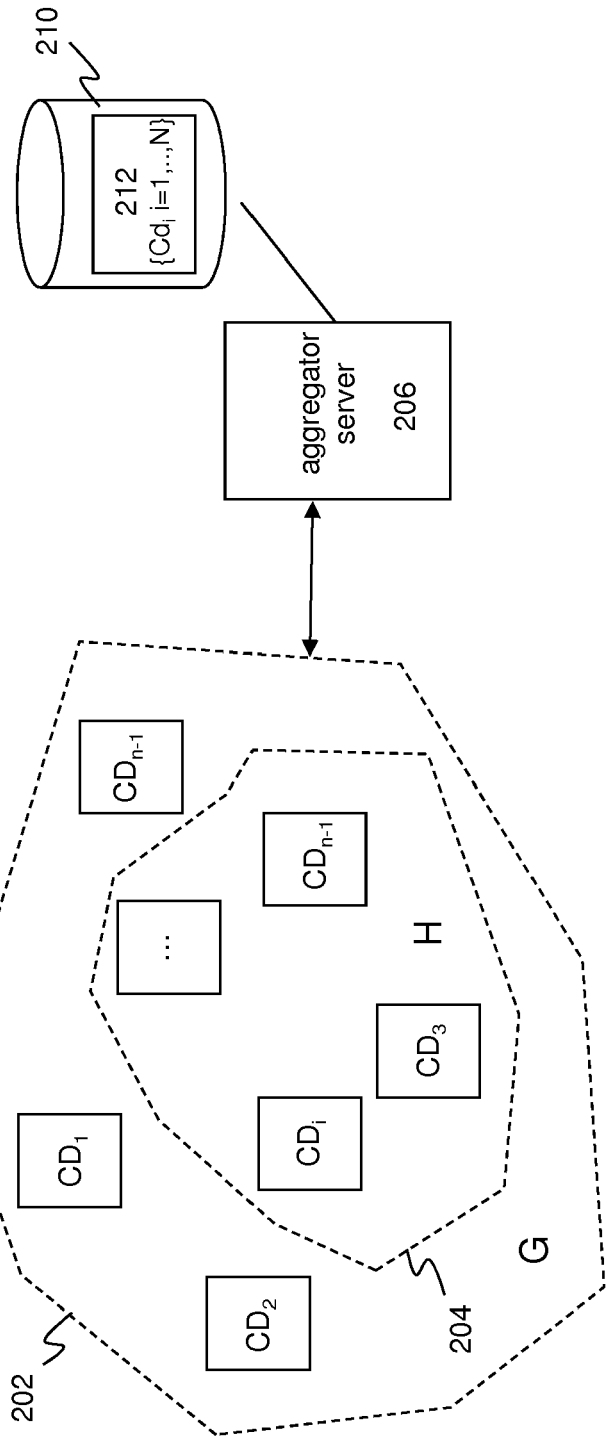
210
212
{Cd_i i=1,..,N}
aggregator server 206
CD_n-1
CD_n-1
...
H
CD_3
CD_i
CD_1
CD_2
202
204
G

FIG. 2

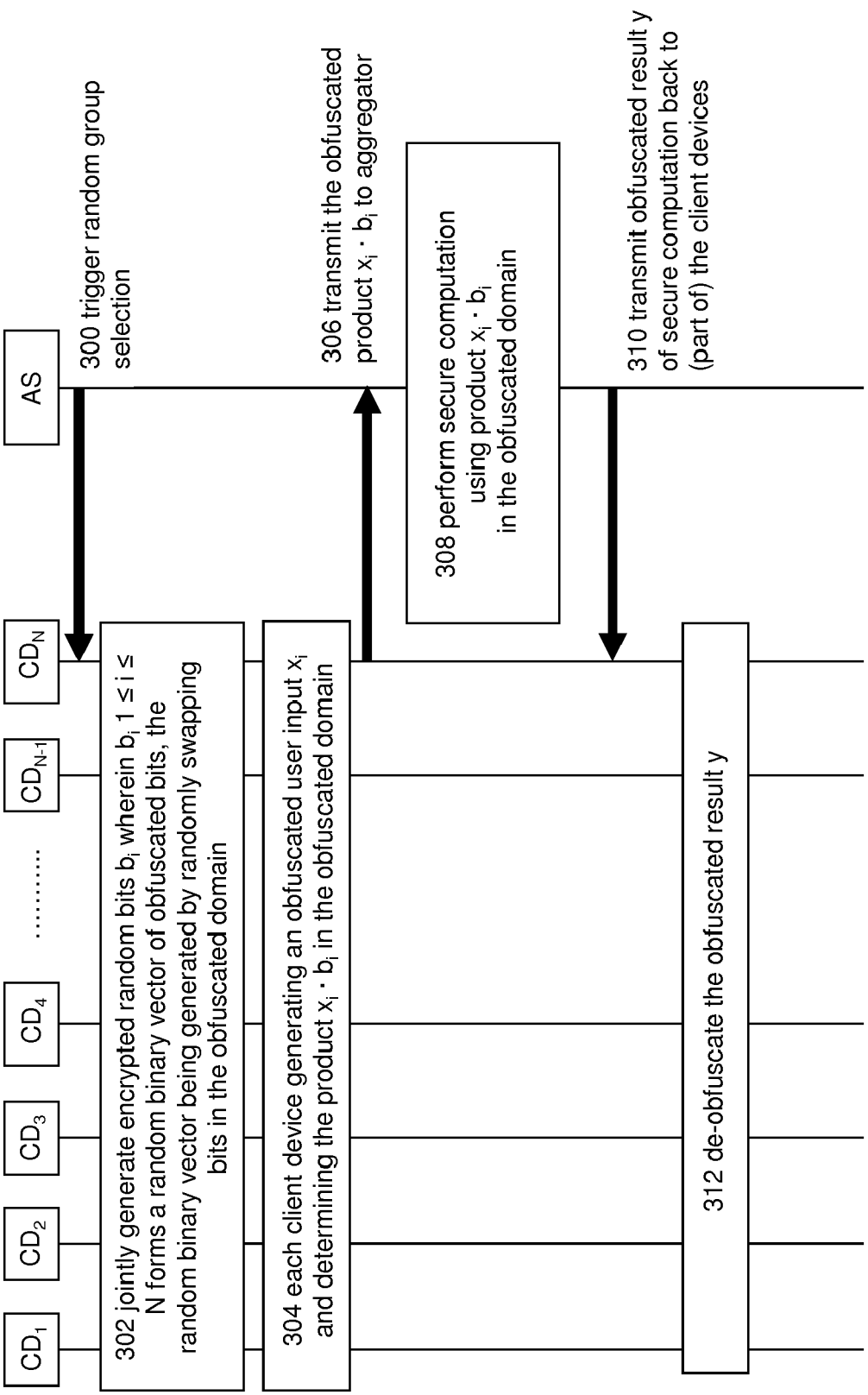
AS
300 trigger random group selection
306 transmit the obfuscated product $x_i \cdot b_i$ to aggregator
308 perform secure computation using product $x_i \cdot b_i$ in the obfuscated domain
310 transmit obfuscated result y of secure computation back to (part of) the client devices
CD_1
CD_2
CD_3
CD_4
.....
CD_{N-1}
CD_N
302 jointly generate encrypted random bits b_i wherein b_i 1 ≤ i ≤ N forms a random binary vector of obfuscated bits, the random binary vector being generated by randomly swapping bits in the obfuscated domain
304 each client device generating an obfuscated user input x_i and determining the product $x_i \cdot b_i$ in the obfuscated domain
312 de-obfuscate the obfuscated result y

FIG. 3

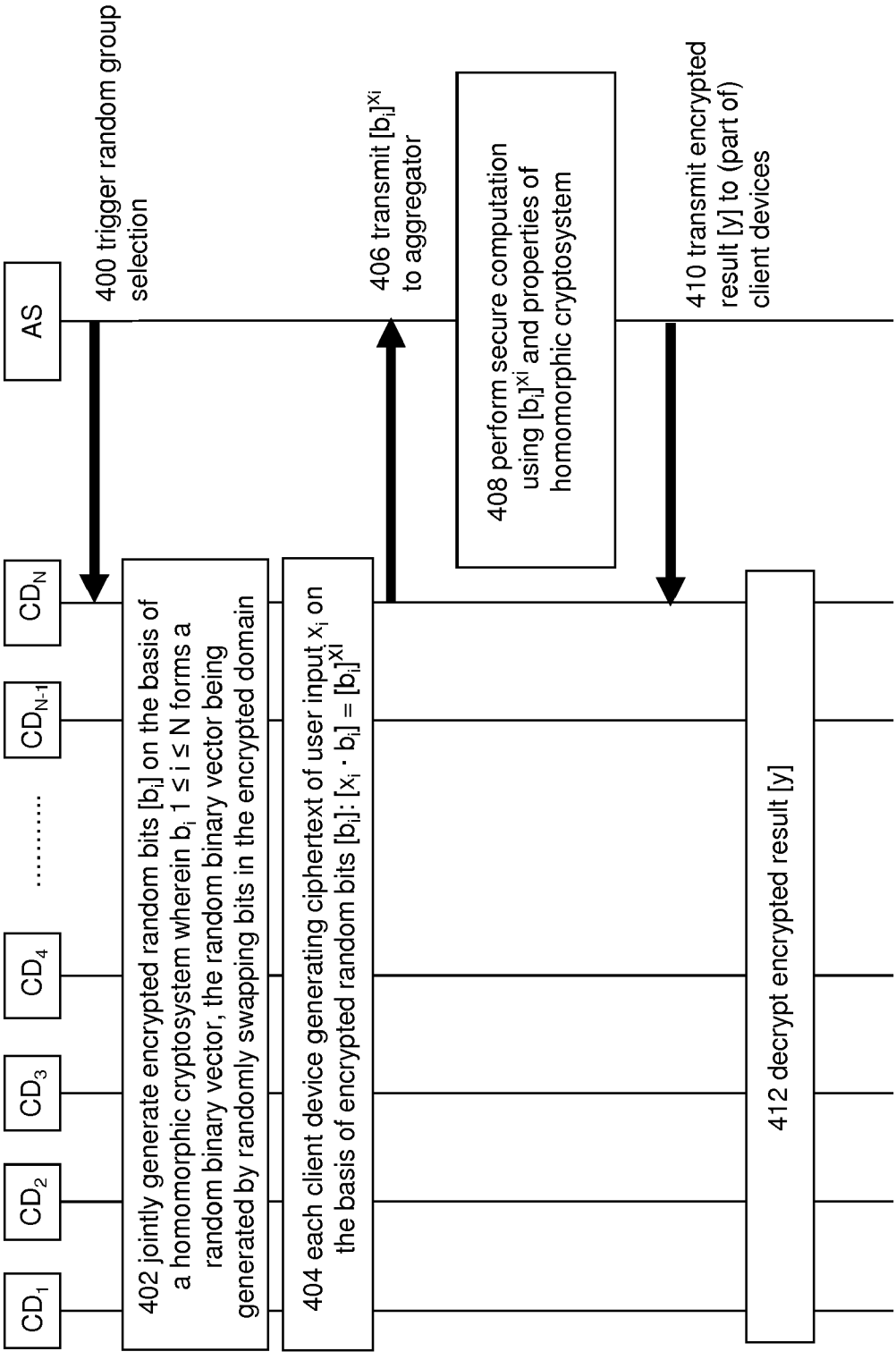
AS
400 trigger random group selection
406 transmit $[b_i]^{x_i}$ to aggregator
408 perform secure computation using $[b_i]^{x_i}$ and properties of homomorphic cryptosystem
410 transmit encrypted result [y] to (part of) client devices
CD_1
CD_2
CD_3
CD_4
.....
CD_{N-1}
CD_N
402 jointly generate encrypted random bits $[b_i]$ on the basis of a homomorphic cryptosystem wherein b_i $1 \leq i \leq N$ forms a random binary vector, the random binary vector being generated by randomly swapping bits in the encrypted domain
404 each client device generating ciphertext of user input x_i on the basis of encrypted random bits $[b_i]$: $[x_i \cdot b_i] = [b_i]^{x_i}$
412 decrypt encrypted result [y]

FIG. 4

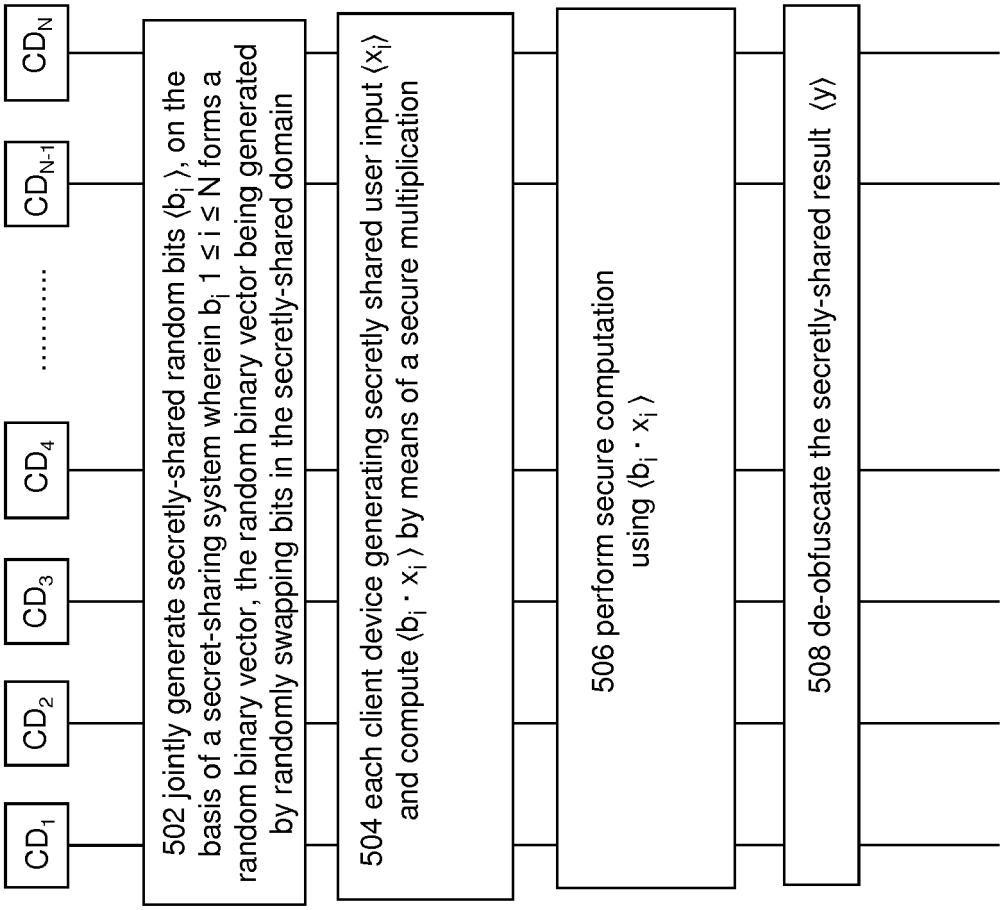
CD1
CD2
CD3
CD4
.....
CDN-1
CDN
502 jointly generate secretly-shared random bits ⟨bi⟩, on the basis of a secret-sharing system wherein bi 1 ≤ i ≤ N forms a random binary vector, the random binary vector being generated by randomly swapping bits in the secretly-shared domain
504 each client device generating secretly shared user input ⟨xi⟩ and compute ⟨bi · xi⟩ by means of a secure multiplication
506 perform secure computation using ⟨bi · xi⟩
508 de-obfuscate the secretly-shared result ⟨y⟩

FIG. 5

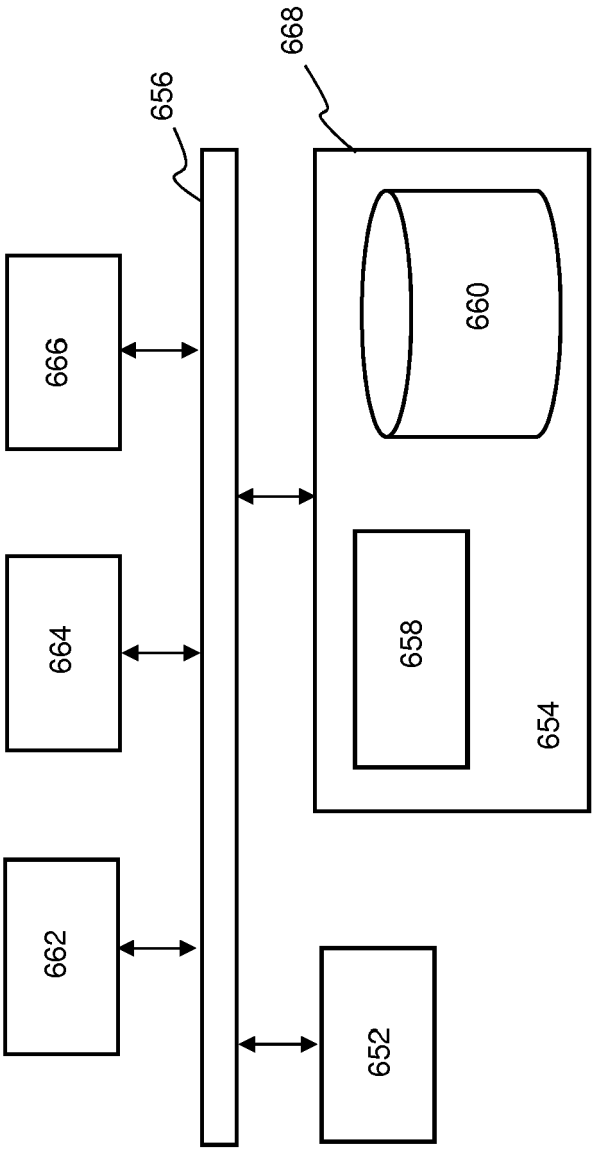

FIG. 6

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPEAN SEARCH REPORT

Application Number
EP 17 21 0886

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X,D	KONONCHUK DMITRY ET AL: "Privacy-Preserving User Data Oriented Services for Groups with Dynamic Participation", 9 September 2013 (2013-09-09), MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION - MICCAI 2015 : 18TH INTERNATIONAL CONFERENCE, MUNICH, GERMANY, OCTOBER 5-9, 2015; PROCEEDINGS; [LECTURE NOTES IN COMPUTER SCIENCE; LECT.NOTES COMPUTER], SPRINGER INTERNATIONAL PUBLISHING, CH, XP047038311, ISSN: 0302-9743 ISBN: 978-3-642-38287-1 * section 2.3; pages 431-433 *	1-15	INV. H04L9/00 H04L9/08
X,D	THIJS VEUGEN ET AL: "Improved privacy of dynamic group services", EURASIP JOURNAL ON INFORMATION SECURITY, BIOMED CENTRAL LTD, LONDON, UK, vol. 2017, no. 1, 1 February 2017 (2017-02-01), pages 1-9, XP021240671, DOI: 10.1186/S13635-017-0054-7 * section 3.1 *	1-15	TECHNICAL FIELDS SEARCHED (IPC) H04L
T	B. R. HEAP: "Permutations by Interchanges", COMPUTER JOURNAL., vol. 6, no. 3, 1 November 1963 (1963-11-01), pages 293-298, XP055504805, GB ISSN: 0010-4620, DOI: 10.1093/comjnl/6.3.293 * the whole document *		

-/--

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	6 September 2018	Manet, Pascal

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPEAN SEARCH REPORT

Application Number
EP 17 21 0886

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
T	DAMGÅRD IVAN ET AL: "Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation", 4 March 2006 (2006-03-04), MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION - MICCAI 2015 : 18TH INTERNATIONAL CONFERENCE, MUNICH, GERMANY, OCTOBER 5-9, 2015; PROCEEDINGS; [LECTURE NOTES IN COMPUTER SCIENCE; LECT.NOTES COMPUTER], SPRINGER INTERNATIONAL PUBLISHING, CH, XP047422636, ISSN: 0302-9743 ISBN: 978-3-540-70543-7 * abstract * * sections 3, 5 and 6 * -----	14,15	
A	MOVAHEDI MAHNUSH ET AL: "Secure Multi-party Shuffling", 20 November 2015 (2015-11-20), MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION - MICCAI 2015 : 18TH INTERNATIONAL CONFERENCE, MUNICH, GERMANY, OCTOBER 5-9, 2015; PROCEEDINGS; [LECTURE NOTES IN COMPUTER SCIENCE; LECT.NOTES COMPUTER], SPRINGER INTERNATIONAL PUBLISHING, CH, XP047413439, ISSN: 0302-9743 ISBN: 978-3-540-70543-7 [retrieved on 2015-11-20] * the whole document * -----	1-15	TECHNICAL FIELDS SEARCHED (IPC)

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	6 September 2018	Manet, Pascal

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- WO 2013066176 A **[0004] [0052]**
- WO 2013066177 A **[0004] [0052]**
- EP 17177653 A **[0046] [0072]**

Non-patent literature cited in the description

- Privacy-preserving user data oriented services for groups with dynamic participation. **KONONCHUK et al.** Computer Security (ESORICS) Lecture Notes in Computer Science. Springer, 2013, 8134 **[0004]**
- **VEUGEN et al.** Improved privacy of dynamic group services. *EURASIP journal on information security,* March 2017 **[0005]**
- threshold cryptosystems. **Y. DESMEDT ; Y. FRANKEL.** CRYPTO. Springer-Verlag, 1989, 308-315 **[0045]**
- Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. **DAMGARD et al.** TCC 2006, LNCS. Springer, 2006, vol. 3876, 285-304 **[0047]**
- Practical Two-Party Computation based on the Conditional Gate. **B. SCHOENMAKERS ; P. TUYLS.** ASIACRYPT 2004, Lecture Notes in Computer Science 3329. Springer-Verlag, 2004, 119136 **[0066]**
- **DAMGARD et al.** *Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation* **[0074]**
- **Z ERKIN et al.** Generating private recommendations efficiently using homomorphic encryption and data packing. *IEEE Trans. Inform. Forensics Secur.,* 2012, vol. 7 (3), 1053-1066 **[0085]**
- **J KACPRZYK.** Group decision making with a fuzzy linguistic majority. *Fuzzy Sets Syst,* 1986, vol. 18, 105-118 **[0085]**
- **F ANSELMI et al.** Unsupervised learning of invariant representations with low sample complexity: the magic of sensory cortex or a new framework for machine learning?. *J. Theor. Comput. Sci.,* 2016, vol. 633 (C), 112-121 **[0085]**
- **A MATHES.** Folksonomies-cooperative classification and communication through shared metadata. *Comput. Mediated Commun.,* 2004, vol. 47, 1-28 **[0085]**
- IEEE International Workshop on Information Forensics and Securit. **Z ERKIN et al.** Privacy-preserving user clustering in a social network. IEEE, 2009 **[0085]**
- On the privacy preserving properties of random data perturbation techniques. **H KARGUPTA et al.** ICDM, IEEE Computer Society. IEEE, 2003, 99-106 **[0085]**