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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 September 2016 (15.09.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/141860 A1

(51) International Patent Classification:
H04L 29/08 (2006.01) *H04L 9/32* (2006.01)

(21) International Application Number:
PCT/CN2016/075668

(22) International Filing Date:
4 March 2016 (04.03.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/130,543 9 March 2015 (09.03.2015) US

(72) Inventor; and
(71) Applicant : **DING, Jintai** [CN/CN]; North Campus 28-203, University of Science and Technology of China, No.96, JinZhai Road, Baohe District, Hefei, Anhui 230026 (CN).

(74) Agent: **BEYOND TALENT PATENT AGENT FIRM**; Room 1202, Kuntai Building, #10 Chaoyangmenwai Str., Chaoyang District, Beijing 100020 (CN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, 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))*

WO 2016/141860 A1

(54) Title: HYBRID FULLY HOMOMORPHIC ENCRYPTION (F.H.E.) SYSTEMS

(57) Abstract: Using a secure hardware or other form secure elements, where we securely implement the decryption and then encryption function to perform the re-encryption function, we build a hybrid fully homomorphic encryption system, where the bootstrap step is replaced the re-encryption function in the hardware module. This new hybrid system are very efficient because the re-encryption is much more efficient than the bootstrap function, which is the main bottleneck in terms of computations in FHE. In such a system, we make the system secure by making this hardware module secure using all or some of known techniques in-cluding temper proof, self-destruction and etc. This module can be controlled by either the server or the client or jointly.

Description

Title: Hybrid fully homomorphic encryption (F.H.E.) systems

Background

[1] This invention is related to the construction of efficient homomorphic encryption systems, in particular, fully homomorphic encryption, where any computation can be performed on encrypted data to protect the secrecy and the privacy of the data.

[2] In our modern information systems, users often have their data stored and managed on large servers or clouds, which they do not have real control, for example, users may store their data in an Amazon cloud. However from the perspective of the users, the secrecy and the privacy of the data becomes a serious concern, since the server has the full control of the data.

[3] One solution to this problem is that the users instead encrypt their data using a symmetric cryptosystem like AES with their own keys and store it in a cloud such that only each user can decrypt the data with their own keys. However this present another problem in the sense that the users can not make full use of the advantage of the powerful computing power to process the data since for a usual encryption like AES we can not perform meaningful operation on encrypted data. This diminishes tremendously the advantage of using the clouds.

[4] Homomorphic encryption is a type of encryption scheme which allows computations over the encrypted data, namely the ciphertext, and derive an encrypted result when decrypted, gives result of computations performed over the plaintext. The feature is very suitable for privacy protection and for cloud computing.

[5] The power of fully homomorphic encryption was recognized within a year of the development of RSA and there are efficient (partially) homomorphic systems, where only certain type of computations on the encrypted data like addition (only) can be performed on the encrypted data.

[6] An idea solution to the problem is to use what is called fully homomorphic encryption (FHE) systems, where any computation can be performed on the encrypted data. Theoretically speaking, a cryptosystem which supports both addition and multiplication on encrypted data is a fully homomorphic encryption (FHE). FHE allows programs to run on encryptions of their inputs to produce an encryption of their output. Since such a program never decrypts its input, therefore it can be run by an untrusted third party without revealing any information on the processed information.

[7] But only after 30 years of the initial suggestion of the idea of homomorphic encryption, recently a number of FHE systems are proposed. The first one was proposed by Craig Gentry in 2009. Craig Gentry used lattice cryptography to build the first FHE system.

[8] Theoretically Gentry's system can provide evaluations of arbitrary depth circuits (any algebraic computations), but his construction starts from a somewhat homomorphic encryption scheme with a limit on essentially the degree of polynomials that can be computed over encrypted data. Then he built a technique called **bootstrap**, which is essentially to actually evaluate its own decryption circuit, to build a fully homomorphic encryption. But this step is very costly practically and therefore the systems is not efficient

[9] There are different variants of Gentry's scheme with smaller key and ciphertext sizes, but still not practical.

[10] There are constructions based on Integers, Learning With Errors problems (LWE) and Ring Learning with Errors problems (RLWE), which are more efficient but again they require bootstrap and the systems are not efficient and not practical.

BRIEF SUMMARY OF THE INVENTION

[11] In this invention, we propose a new paradigm to deal with such a problem using secure hardware or other form of secure elements(SE).

[12] There are already suggestion to use hardware security to achieve the protection of data in cloud while still can perform computations. But in this case, it has a very high demand on the hardware and therefore can be very costly. One such suggestion is by Ken Eguro and Ramarathnam Venkatesan of Microsoft (FPGAS FOR TRUSTED CLOUD COMPUTING).

[13] In our new proposal, we suggest a totally new paradigm, namely we propose a hybrid approach. We combine both the idea of hardware security with the HFE.

[14] We propose to add a secure and efficient bootstrap module to a FHE, which are based either on hardware to other form of secure elements (SE), to the system to perform the function of the bootstrap step, but we will not use the bootstrap computations, but rather we use **the secret key of the users** to perform the decryption and then re-encrypt the data as the out-put.

[15] In our case, this step must be performed in a secure hardware or other from of secure elements, where the secret key of the user must be fully protected.

[16] Our usage of hardware is very different from another direction of usage of hardware, namely there are lots of efforts to develop hardware to speed the bootstrap computations itself, but we use direct decryption and re-encryption to efficiently perform the function of bootstrap.

DETAILED DESCRIPTION OF THE INVENTION

1.1 The basic idea of FHE and bootstrap

[17] For a FHE system, the data is divided in small blocks of fixed sizes, which we will denoted an x_i, $i \in N$, natural numbers. There are addition and multiplication defined on these block, namely we can do addition and multiplication on the blocks: $x_i + x_j$, $x_i \times x_j$.

[18] The system allows each user an encryption function and a decryption function , and the encryption function has a public key P, and we denote this function as E_P, and the decryption function has a secret key S, which only the user knows, and we will denote as D_S. These two function are very efficient in general.

[19] For any data block, which we also call a plaintext we apply encryption to derive an encrypted block:

$$y_i = E_P(x_i).$$

y_i can have different size from x_i.

[20] We also have addition and multiplication on the encrypted blocks: $y_i + y_j$, $y_i \times y_j$. These addition and multiplication in general are not the same as that of the plaintext blocks.

[21] For such an homomorphic encryption system, we in general have the homomorphic property:

$$D_S(y_i + y_j) = x_i + x_j,$$

$$D_S(y_i \times y_j) = x_i \times x_j,$$

[22] In general, for a plaintext x_i, there are many ciphertexts, namely there are other $y' \neq y_i$ such that

$$D_S(y') = x_i.$$

[23] The reason for the situation above is that the encryption process allows certain errors to be added and as long as the error is within certain range, we will decryption correctly. But If it is out of the limit, the decryption will fail to give the desire results.

[24] The error is enlarged once we start to do computations on the encrypted data y_i and if we do too many operations in particular, multiplications, the error will be out of control.

[25] Bootstrap is a solution to this problem, and it is essentially to evaluate its own decryption circuit. What is does is to refresh the ciphertext so that the error refreshed to the original level. W denote this function as B_S: $B_s(y")$ will be decrypted to the same plaintext but it should has the same level of error terms as $y)i$ and should have much smaller error term than $y"$.

[26] To have a true FHE, we must have B_S, it is difficult to implement and it is very very inefficient. This is one of the main reasons why we could use the FHE in cloud computing in large scale yet.

1.3 A hybrid FHE construction

[27] To illustrate the point in a clear way, we will explain our construction in a setting of cloud computation. Surely our construction can be applied to broad and similar applications.

[28] We have a cloud server and a client using this server. As we described above, the client first encrypt all it data x_i using it own public key (or it can keep it private) and its encryption functions E_P:

$$y_i = E_P(x_i).$$

[29] Then the client will put the encrypted data y_i on the cloud.

[30] In addition, the client can provide a secure hardware, which serves as a bootstrap machine, name, this hardware has only one function, given any input, it will decrypt it using D_S and then it will re-encrypt it using E_P and give the output as a the re-encrypted message.

[31] This secure hardware allows one functionality only given input and give an output, nothing else. We will denote this function as RE_S.

[32] Then the client will provide this hardware to the cloud and it will be integrated in the cloud server, but this device will be kept in a very secure area in the cloud server.

[33] The key point of the hybrid system is that when the cloud sever need to do any computation on the encrypted data in the cloud, and when they need do a bootstrap step

to perform the function B_K in the original FHE, they will just call this new hardware to do the computation of RE_S. This will solve the problem of efficiency of bootstrap.

[34] When the client or anyone wants to compute the value of any algebraic function $f(x_1, ..., x_N)$ for any fixed integer N, it will give the function $f(x_1, ..., x_N)$ to the cloud server, and the cloud server will compute $f(y_1, ..., y_N)$ using the homomorphic property of E_P, in addition, during this process whenever there is a need of using the function B_S in the original FHE, the server will call the secure module to apply the function RE_S to perform the re-encryption and refresh the errors to the original level from encryption by using E_P.

[35] When this result of $f(y_1, ..., y_N)$ is derived, it will be sent to the client, who can decrypt it if the client wants to, or ignore it if it does not want to.

[36] The advantage compared to the system supported by hardware is that here we only need to protect a small piece of hardware to be secure not a large systems, and therefore it is will be very costly effective.

[37] We will use all (or some) tools to make this hardware secure to protect secrecy of D_S and S, which include:

(1) it has a power supply to protect it from tempering and if any tempering is detect, it will wipe out the whole program;

(2) it has a temper detection circuit to send out warning to the client and the cloud server;

(3) the secret keys are impossible to find even if some gets hold of the hardware module.

[38] This hardware is small since the decryption is very easy and it is fast. Therefore it is of low cost and it is easy to make it secure since it is a small device.

[39] We can also keep this hardware at the client site but it has a fast connection to the cloud server and the client allows the cloud server fast access to this module of the functionality of RE_S.

[40] This secure hardware can be part of the service provided and even controlled by the cloud server. Again, the advantage is that we only need to protect a small piece hardware not a large system, therefore it is very practical.

[41] we can also use other form of secure elements like TrustZone etc to implement the function RE_S, as long it is secure and of low cost.

[42] We can also replace the secure hardware module with a secure software either in the white-box form or in an obfuscated software, where the decryption part (or the key) are fully protected.

[43] Our construction can be illustrated in the figure below.

LITERATURE CITED

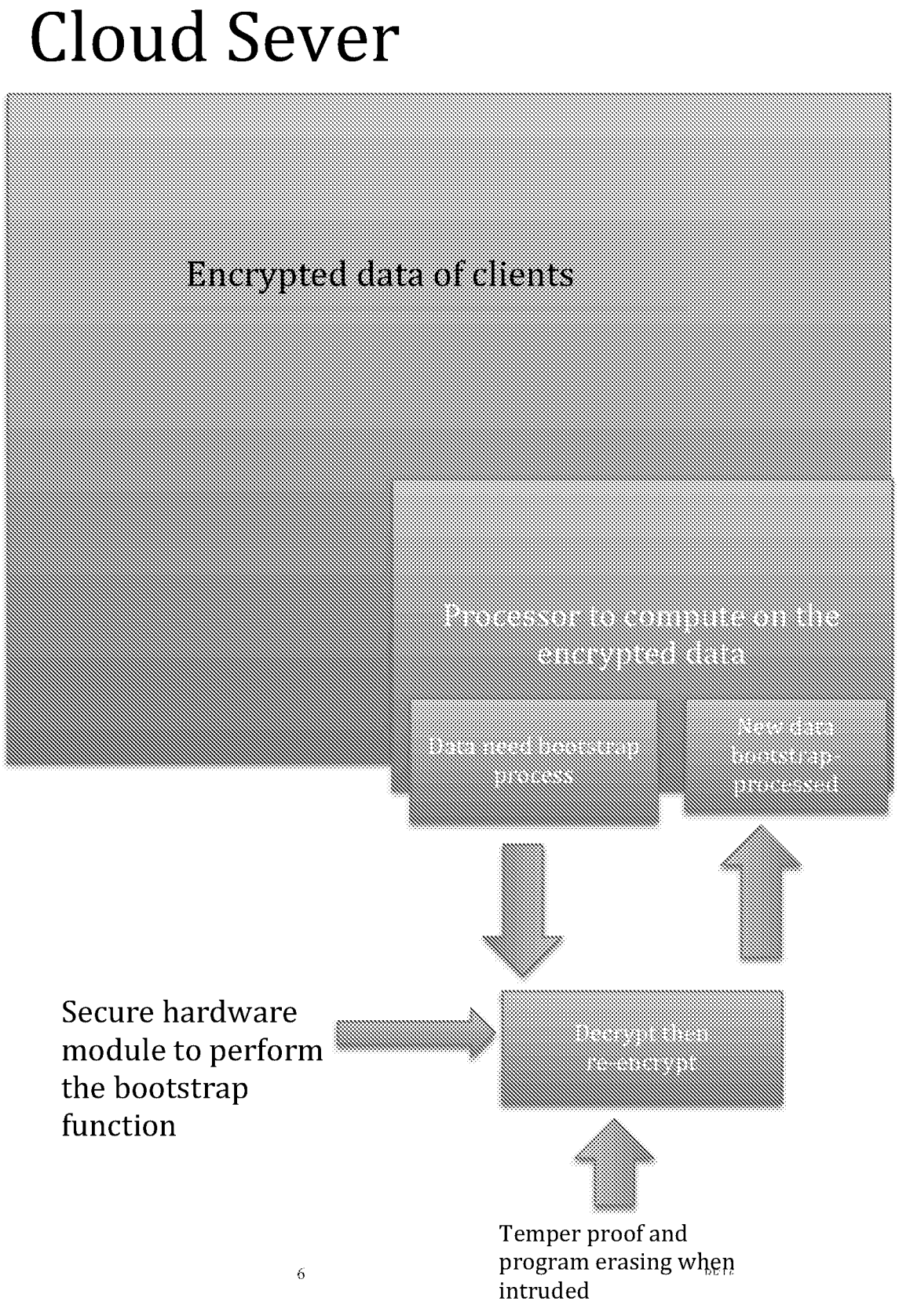
Craig Gentry, Fully homomorphic encryption using ideal lattices, Symposium on the Theory of Computing (STOC), 2009, pp. 169-178.

Z. Brakerski and V. Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. In FOCS 2011 (IEEE)

Marten, van Dijk; Gentry, Craig; Halevi, Shai; Vinod, Vaikuntanathan. "Fully Homomorphic Encryption over the Integers". EUROCRYPT 2010 (Springer).

Zvika Brakerski and Vinod Vaikuntanathan Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages CRYPTO 2011.

Ken Eguro and Ramarathnam Venkatesan, FPGAs for Trusted Cloud Computing, in International Conference on Field-Programmable Logic and Applications, IEEE, August 2012.


Cloud Sever
Encrypted data of clients
Processor to compute on the encrypted data
Data need bootstrap process
New data bootstrap processed
Secure hardware module to perform the bootstrap function
Decrypt then re-encrypt
Temper proof and program erasing when intruded

CLAIM

[1] Claim 1. Method for establishing a hybrid fully homomorphic encryption scheme, comprising:

(1) *Each user or client has a FHE system set up, where it has an encryption function and a decryption function , and the encryption function has a public key P, and we denote this function as E_P, and the decryption function has a secret key S, which only the user knows, and we will denote as D_S. Each user also has bootstrap function denoted as B_S, where this function is essentially to evaluate its own decryption circuit to refresh a ciphertext so that the error inside is refreshed to the original level when a plaintext is encrypted by the encryption function E_P. For any ciphertext y", $B_s(y")$ will be decrypted to the same plaintext but it should has the same level of error terms as a ciphertext derived from applying directly the encryption function.*

(2) *We then set up a hybrid system in the the following way:*

(a) *each user or client will first encrypt all it data blocks x_i, i natural numbers using it own public key (or it can keep it private) and its encryption functions E_P:*

$$y_i = E_P(x_i),$$

and then store the data in a cloud server or similar server. It will provide the encryption E_P too the cloud (and the cloud can keep it private to public).

(b) *each client provides a secure hardware, which serves as a bootstrap machine, name, this hardware has only one function, given any input, it will decrypt it using D_S and then it will re-encrypt it using E_P and give the output as a the re-encrypted message. This function is denoted as RE_S.*

(3) *When the client or anyone wants to compute the value of any algebraic function $f(x_1, ..., x_N)$ for any fixed integer N, it will give the function $f(x_1, ..., x_N)$ to the cloud server, and the cloud server will compute $f(y_1, ..., y_N)$ using the homomorphic property of E_P, in addition, during this process whenever there is a need of using the function B_S in the original FHE, the server will call the secure module to apply the function RE_S to perform the re-encryption and refresh the errors to the original level from encryption by using E_P.*

(4) *When this result of $f(y_1, ..., y_N)$ is derived, it will be sent to the client, who can decrypt it if the client wants to, or ignore it if it does not want to.*

[2] Claim 2. We use the client and cloud server setting to illustrate how our hybrid system works in a clear way, but our system is applicable to broad and similar applications.

[3] Claim 3. The secure hardware module to perform only one functionality, namely given any input and give an output, nothing else. .

[4] Claim 4. The client will provide the hardware module, or the client will be the one configure a hardware withs secret key and provide to the cloud and it will be integrated in the cloud server, but this device will be kept in a very secure area in the cloud server.

[5] Claim 5. The client and the server will use all (or some) tools to make this hardware secure to protect the secrecy of D_S and S, which include:

(1) it has a power supply to protect it from tempering and if any tempering is detect, it will wipe out the whole program;
(2) it has a temper detection circuit to send out warning to the client and the cloud server;
(3) the secret keys are impossible to find even if some gets hold of the hardware module.

[6] Claim 6. The client can also keep this hardware module at the client site but it has a fast connection to the cloud server and allow the cloud server fast access to this module to use the function RE_S.

[7] Claim 7. This secure hardware can be part of the service provided and even controlled by the cloud server if the client trust server to maintain the secrecy of D_S and S. In this case, the server will build special hardware to securely implement RE_S and to maintain the security of D_S and S

[8] Claim 8. Clients or servers can also use other form of secure elements like TrustZone etc to implement the function RE_S, as long it is secure and of low cost.

[9] Claim 9. Clients or servers also replace the secure hardware module with a secure software either in the white-box form or in an obfuscated software, where the decryption part (or the key) are fully protected.

[10] Claim 10. The hardware module can be controlled or built by the server only, or the the client only or jointly depending on the needs of the clients.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2016/075668

A. CLASSIFICATION OF SUBJECT MATTER

H04L 29/08(2006.01)i; H04L 9/32(2006.01)i

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)

H04L

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)

CNPAT, WPI, EPODOC, USTXT, EPTXT, WOTXT, CNKI, GOOGLE, IEEE: re-encrypt+, encrypt+, decrypt, fully, homomorphic, homomorphical, FHE, cloud, key, bootstrap, error, secret, hybrid, again, second

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
PX	CN 104426973 A (CHINA MOBILE COMMUNICATIONS CORPORATION) 18 March 2015 (2015-03-18) description, paragraphs[0042]-[0063], [0101]-[0106]	1-10
A	US 8515058 B1 (THE BOARD OF TRUSTEES OF THE LELAND STANFORD JUNIOR UNIVERSITY) 20 August 2013 (2013-08-20) description, column 37, paragraph 2 to column 47, paragraph 6, claims 1-16	1-10
A	US 2014095860 A1 (ALCATEL- LUCENT USA INC. ET AL.) 03 April 2014 (2014-04-03) the whole document	1-10
A	WO 2012149395 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 01 November 2012 (2012-11-01) the whole document	1-10
A	CN 104283669 A (SOUTHEAST UNIVERSITY) 14 January 2015 (2015-01-14) the whole document	1-10

☑ 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
09 May 2016	**27 May 2016**

Name and mailing address of the ISA/CN	Authorized officer
STATE INTELLECTUAL PROPERTY OFFICE OF THE P.R.CHINA **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088, China**	**LI,Dandan**
Facsimile No. **(86-10)62019451**	Telephone No. **(86-10)62413305**

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2016/075668

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	SMART, Nigel P. et al. "Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes" *Public Key Cryptography – PKC 2010, Lecture Notes in Computer Science,* Vol. 6056, 31 December 2010 (2010-12-31), pages 420-443	1-10
A	PORRAS, Jaiberth et al. "ZHFE, a New Multivariate Public Key Encryption Scheme" *POST-QUANTUM CRYPTOGRAPHY, PQCRYPTO 2014, Lecture Notes in Computer Science,* Vol. 8772, 31 December 2014 (2014-12-31), pages 229-245	1-10

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2016/075668

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 104426973 A	18 March 2015	None	
US 8515058 B1	20 August 2013	None	
US 2014095860 A1	03 April 2014	None	
WO 2012149395 A1	01 November 2012	US 2015358153 A1	10 December 2015
		US 2013170640 A1	04 July 2013
CN 104283669 A	14 January 2015	None	