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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 February 2014 (20.02.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/028194 A1

(51) International Patent Classification:
G06F 21/30 (2013.01) *G06F 21/62* (2013.01)

(21) International Application Number:
PCT/US2013/052194

(22) International Filing Date:
26 July 2013 (26.07.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/682,385 13 August 2012 (13.08.2012) US

(71) Applicant *(for all designated States except US)*: **SIEMENS CORPORATION** [US/US]; 170 Wood Avenue South, Iselin, New Jersey 08830 (US).

(72) Inventor; and

(71) Applicant *(for US only)*: **GRAVEMAN, Richard F.** [US/US]; 15 Park Avenue, Morristown, New Jersey 07960 (US).

(74) Agents: **PASCHBURG, Donald B.** et al.; Siemens Corporation- Intellectual Property Dept., 170 Wood Avenue South, Iselin, New Jersey 08830 (US).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,

[Continued on next page]

(54) Title: SECURELY GENERATING AND STORING PASSWORDS IN A COMPUTER SYSTEM

(57) Abstract: Methods and systems for protecting a password are disclosed. According to one aspect of the present invention, a processor selects a set of local configuration data. This can include one or more strings associated with local configuration data. The processor concatenates the set of local configuration data and calculates a hash value of the concatenated data. The processor generates an encrypted string by using the hash value as a key to encrypt the password. Then the processor encodes the encrypted string as a string in a software program. When the password is needed by a first computer system to access a second computer system, the steps are reversed, the password obtained and the first computer system accesses the second computer system

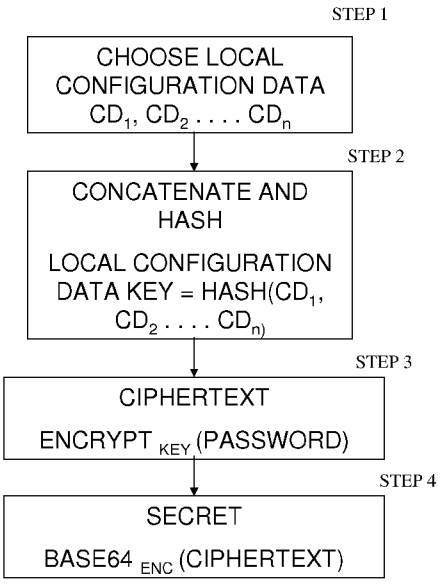

FIG. 7

WO 2014/028194 A1

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

— *with amended claims (Art. 19(1))*

Published:

— *with international search report (Art. 21(3))*

SECURELY GENERATING AND STORING PASSWORDS IN A COMPUTER SYSTEM

STATEMENT OF RELATED CASES

[0001] The present application claims priority to and the benefit of U.S. Provisional Patent Application Serial No. 61/652,355 filed on August 13, 2012, which is incorporated herein by reference in its entirety.

BACKGROUND

[0002] Securing passwords and other information is important to protect the security of computers. Passwords are a key defense to unwanted intrusion into a computer system. The loss of a password, often through hacking, and the resulting loss of secure information, is often disastrous, as important and highly sensitive data can be lost.

[0003] Password security issues exists even when computer systems interact. In those cases, the first computer system accessing a second computer system must still access the second computer system via a password.

[0004] When a first system automatically accesses a second system, passwords (or other authentication information, called simply passwords herein) may need to be stored in the first system rather than being entered by a person. This often increases the risk that these passwords may be acquired by anyone having access to the executable software or configuration files containing such passwords.

[0005] To reduce this risk, new and improved systems and methods to generate, store and access passwords are needed.

SUMMARY OF THE INVENTION

[0006] The present invention processes the password using local configuration data, data concatenation, hash functions and encryption to protect passwords. This information can be stored on a first computer device. The information can be stored in memory on the first computer device as software applications. The information can also be stored in configuration files in memory on the first computer device. When a first computing device needs to access a second computing device, the first computing device reverses the process to obtain the password.

[0007] One aspect of the present invention is a method of protecting a password. A processor, under control of an instruction set in memory, selects a set of local configuration data, concatenates the set of local configuration data and calculates a hash value of the concatenated data. The processor generates an encrypted string by using the hash value as a key to encrypt the password and then encodes the encrypted string as a string in a software program.

[0008] In accordance with further aspects of the present invention, a SHA-256 hash value is calculated. Other hash values can also be calculated.

[0009] In accordance with further aspects of the present invention, an AES-256 key can be used to perform the encryption step.

[0010] The encryption step can also use ECB Mode or Counter Mode. Other key sizes, encryption methods, or modes of encryption can also be used.

[0011] The encoding step, in accordance with another aspect of the present invention uses Base 64 to perform the encoding step, although other encoding processes or none at all at this step can also be used.

[0012] The local configuration data can be constructed using different information and different processes. In accordance with one embodiment of the present invention, a CPUID instruction from the processor is used to generate a local configuration data. In accordance with another embodiment of the present invention, a 48-bit MAC address from a network interface card is used to generate the local configuration data. In a further embodiment, a software license number from an operating system is used to generate the local configuration data. In yet another embodiment, a model and serial number from a peripheral device connected to the processor is used to generate the local configuration data. These local configuration data may also be used in combination.

[0013] In accordance with further aspects of the present invention, the processor also performs the following steps when its computer system wants to access another computer system. When it needs to reconstruct the password, the processor decodes the encrypted string to get binary ciphertext, obtains the set of local configuration data and calculates the hash value to obtain a reconstructed key and then decrypts the binary ciphertext with the reconstructed key to obtain the password. Then the processor uses the password to access a computer system.

[0014] After the password has been reconstructed, the processor preferably destroys the reconstructed key. In one embodiment of the present invention, the processor erases the reconstructed key in memory. In another embodiment of the present invention, the processor writes over the reconstructed key in memory to destroy the reconstructed key.

[0015] The present invention also contemplates a system for protecting passwords. The system is a computer system that has a memory storing an instruction set and a processor in communication with the instruction set in the memory. The instruction set is operable to cause the processor to: select a set of local configuration data; concatenate the set of local configuration data and calculate a hash value of the concatenated data; generate an encrypted string by using the hash value as a key to encrypt the password; and encode the encrypted password and store the encrypted password in the memory.

[0016] In one embodiment of the present invention, the encrypted password is encoded in a software application. In another embodiment of the present invention, the encrypted password is stored in a configuration file.

[0017] When the processor needs access to the password, some time later, it decodes the encrypted string to get binary ciphertext, obtains the set of local configuration data and calculates the hash value to obtain a reconstructed key, decrypts the binary ciphertext with the reconstructed key to obtain the password, and uses the password to access a second computer.

DESCRIPTION OF DRAWINGS

[0018] FIG. 1 illustrates storage of a password in a software application, in accordance with an aspect of the present invention.

[0019] FIG. 2 illustrates storage of a password in a configuration file, in accordance with an aspect of the present invention.

[0020] FIG. 3 illustrates a password submission protected by SSL or by TLS, in accordance with an aspect of the present invention.

[0021] FIG. 4 illustrates a protected password in a configuration file, in accordance with an aspect of the present invention.

[0022] FIG. 5 illustrates an obfuscated password, in accordance with an aspect of the present invention.

[0023] FIG. 6 illustrates a password protected by local configuration data, in accordance with an aspect of the present invention.

[0024] FIG. 7 illustrates the steps in creating and protecting a password in accordance with an aspect of the present invention.

[0025] FIG. 8 illustrates the steps in decoding a password in accordance with an aspect of the present invention.

[0026] FIG. 9 illustrates a system in accordance with an aspect of the present invention.

DESCRIPTION

[0027] The goal of this invention is to provide methods and systems to enhance the protection of passwords and other secret symmetric keys, private keys or authentication data. The passwords can be store in software as shown in FIG. 1. In this case, the passwords can be encrypted into a string which is stored as part of a file, a software application or other software program. The passwords can also be stored in configuration files, as show in FIG. 2.

[0028] Passwords stored this way are subject to significant threats. An attacker may access, copy, and analyze software executables or configuration files that are stored on local media. This may occur because of an inside attack, a gap in the access controls protecting the software or configuration files, or access to backup copies of the software or configuration files stored elsewhere.

[0029] The attacker may access, copy, and analyze software executables or configuration files that are stored on remote media (e.g., in a distributed computing or cloud computing configuration).

[0030] The attacker may access, copy, and analyze software as images of running processes in main memory or on a paging device.

[0031] The attacker may access, copy and analyze copies of the software written after system crashes or full or partial copies of software obtained by side channel attacks or found in storage devices including DRAM after the power has been shut down.

[0032] Generally, the model described in this invention assumes that the attacker may *not* access, copy, or trace the running software during the operations where the password is used. Also, generally, the attacked may *not* simply steal the password as it is sent from the first system to the second.

[0033] Methods exist for protecting passwords. Methods may provide for (1) controlling access to software executable files and configuration files (see FIG. 4); or (2) code obfuscation to make reverse engineering software and extracting passwords or other secret information

difficult (see FIG. 5). Additionally, as shown in FIG. 3, a password submission can be protected by SSL or by TLS, in accordance with an aspect of the present invention. Also, a password can be protected within a configuration file, as shown in FIG. 6. These measures can be used, but they are preferably used in conjunction with the present invention, because, while they may be helpful, they may not be sufficient in all scenarios. Of course, a password may be encrypted, but this may simply shift the problem from protecting the password to protecting the decryption key.

[0034] The attacker may succeed in obtaining software executable files or configuration files without, however, having full access to the first system or full knowledge about how the software works. Also, the attacker in some cases may have limited, one-time access to the first system and not be able to intrude again to obtain additional information.

[0035] In addition to using whatever access controls and obfuscation methods are available, the present invention makes accessing the password (or other secret authentication data) stored in the software or configuration files of the first system depend on other system parameters that may be unavailable to the attacker or which the attacker with limited access has not obtained or cannot obtain.

[0036] When software is first installed, or when the password changes, some system management operations are needed to insert the password or other secret authentication information into the local system. If, in accordance with an aspect of the present invention, this information depends in a secure way on local configuration data that the attacker may not have, then possessing copies of the software and configuration files alone may be useless for obtaining the password or secret authentication information, and, in addition, running the software elsewhere will not allow the correct password or secret authentication information to be reconstructed, extracted or used.

[0037] The challenge faced by the present invention was to find some information specific to the local configuration that is stabile and repeatable but not easy to guess and not stored in the application software or its configuration files. Some candidates follow:

[0038] Modern CPUs from Intel and AMD have a CPUID instruction. By setting several values of a parameter and executing this instruction repeatedly, one obtains an array of information that, while not unique, may be difficult to guess precisely.

[0039] Network interface cards have a 48-bit MAC address that is globally unique. Data communications on the LAN expose this address but routed communications with other networks do not.

[0040] Operating system software may have a software license number that can be accessed.

[0041] Peripheral devices such as hard drives may have model and serial numbers or configuration information (such as interrupt vectors, addresses or priorities) accessible by software.

[0042] Special-purpose hardware can include a physical random function that returns consistent but externally unpredictable values for some constant inputs.

[0043] In addition to other existing protections, the following process can be used to reduce the likelihood that an attacker can extract passwords or other secret authentication data in the threat model described above. At installation, when the secret authentication information changes, or when the local configuration data change, the following steps, shown in FIG. 7, are performed in accordance with aspects of the present invention:

[0044] Step 1 - Choose a set of local configuration data. The configuration data defined above, can be used. Other configuration data can also be used. It is preferred that the selected local configuration data maximize the chances of being repeatable and are not easily guessable.

[0045] Step 2 - Concatenate the local configuration data and calculate a hash of the concatenated data. The SHA-256 hash can be calculated in accordance with a preferred embodiment. Other methods of processing the local configuration data, such as encrypting the data with a fixed and known key, may also be used.

[0046] Step 3 - Use the hash value as the AES-256 key to encrypt the password or other secret authentication information. For small amounts of secret authentication information, ECB Mode is appropriate. For larger amounts, Counter Mode should be used. Other methods of using the processed local configuration data to encrypt the password, for example, a simple exclusive or (XOR) operation, may be used.

[0047] Step 4 - Store the encrypted password in memory. Preferably, the encryption is encoded as a printable string in the software. The encoding can be Base64, for example, but other encoding can also be used. It is also preferred to apply available and prudent obfuscation to this string.

[0048] When the password is needed, the following steps, illustrated in FIG. 8, are preferably performed by the processor.

[0049] Step 1 - Reverse the printable encoding to get the binary ciphertext.

[0050] Step 2 - Obtain the local configuration data and compute their hash to reconstruct the key. Securely erase (overwrite) the local configuration data.

[0051] Step 3 - Decrypt to obtain the password or secret authentication information and securely erase (overwrite) the key. If only a portion of the secret authentication information is needed, then only the corresponding part of the ciphertext should be decrypted.

[0052] Step 4 - Use the decrypted information and securely erase (overwrite) it.

[0053] In accordance with an aspect of the invention, these steps are performed only at the time the password is needed.

[0054] Variations and extensions are possible.

[0055] This method can be extended easily when more than one password or type of secret authentication information needs to be stored. The individual passwords or other secret data items should be aligned and encrypted with an appropriate mode so that they can be decrypted separately when needed.

[0056] Different hash functions and encryption methods ma be used. SHA-2 and AES-256 are a natural fit but only an example. The purpose of the hash function is to extract uniform pseudo-random bits from the local configuration data. If the password is not too long, then it can simply be exclusive ORed with the output of the hash function. If the hash function is needed more than once, a counter can be concatenated to the local configuration data. In fact, a block cipher such as AES is not needed at all. It is only included to provide a pseudo-random function. Other types of randomness extraction and masking are well known to cryptographers. Another simple and good approach is Krawczyk's extract-and-expand construction. Because this uses only a hash function and no block cipher, it may be subject to fewer export restrictions. (See http://webee.technion.ac.il/~hugo/kdf/kdf.pdf).

[0057] It may be difficult to guarantee that the local configuration data can be reconstructed exactly. The following extensions to the above steps may be used. At installation, construct an error correcting code such that the local configuration data constitute a codeword. When the local configuration data are later retrieved and reconstructed, apply the error correcting code and hope that they decode to the original codeword. Additional

redundancy can be used to help determine whether this step was successful. If it can be determined how the local configuration data have changed, then it may be possible to "re-center" the error correcting code around the "new" local configuration data, to reconfigure the system automatically, and to adapt to a sequence of changes in the local configuration data. Note that this added redundancy may not be free. The error correcting code may help the attacker to reverse engineer the local configuration data. For more details about this process, the research literature on "fuzzy extractors" should be consulted. Note, however, that the concept of a "robust fuzzy extractor" is not needed in this application. In accordance with an aspect of the present invention, list decoding may be a good choice here, because the application can tolerate trying to decode correctly more than once.

[0058] Various alternative choices of local configuration data may exist. The security requirement is characterized by min-entropy. The application and configuration of the first system should be considered when choosing what to use.

[0059] Systems may have more than one MAC address, so a method is needed to choose one or even to use more than one.

[0060] The reasons for the Base64 or other encoding is to avoid storing random binary data that is easily found by reverse engineering searches.

[0061] It may be advisable not to make the crypto too efficient. In fact, slowing it down artificially may be advantageous. If the attacker needs 100,000,000 guesses to find the local configuration data and the crypto takes 1 second, the system is quite secure. If the crypto takes 1 millisecond, it is much less so.

[0062] The actual transmission of the password from the first system to the second should, if possible, be protected by additional means such as SSL-TLS. This is illustrated in FIG. 3. Of course, the SSL-TLS connection needs to be authenticated, but not necessarily by sending a password.

[0063] FIG. 9 illustrates a system used in accordance with an aspect of the present invention. The system includes a processor 200 in communication with memory 202, network interface cards 204, peripheral devices 206 and special purpose hardware 208. The memory stores many different types of information, including operating system software, software applications and programs and an instruction set to cause the processor 200 to perform a number of steps, including those described herein and shown in FIGS. 7 and 8. The software

programs in the memory 202 typically have configuration words associated with them that can be used to implement aspects of the present invention, as described herein.

[0064] The peripheral devices 206 include a wide range of devices, including but not limited to, hard drives and printers. These devices 206 also have configuration words associated with them that can be used to implement aspects of the present invention, as described herein. Likewise, the network interface cards 204 and the special purpose hardware 208 also have configuration words associated with them that can be used to implement aspects of the present invention, as described herein. Essentially, any hardware or software connected to the processor 200 that has configuration words of any type can be used to implement aspects of the present invention described herein.

[0065] When passwords are used in automatic systems, there is not requirement that they have human mnemonic properties or other such restrictions. Avoiding any such artificial restrictions may strengthen the method described.

[0066] Other approaches to this problem based on a single sign-on technology or secure hardware such as trusted platform systems are somewhat different and possibly complementary with this approach.

[0067] While there have been shown, described and pointed out fundamental novel features of the invention as applied to preferred embodiments thereof, it will be understood that various omissions and substitutions and changes in the form and details of the methods and systems illustrated and in its operation may be made by those skilled in the art without departing from the spirit of the invention. It is the intention, therefore, to be limited only as indicated by the claims.

CLAIMS

1. A method of protecting a password in a computer, comprising a processor:

selecting a first local configuration word associated with the computer as a data word;

applying a hash function to the data word to calculate a hash value; and

generating an encrypted string by using the hash value as a key to encrypt the password; and

storing the encrypted string in a memory in the computer.

2. The method of claim 1, comprising the processor, encoding the encrypted string as a string in a software program which is stored in the memory.

3. The method of claim 1, wherein the processor concatenates a second local configuration word with the first local configuration word to generate the data word.

4. The method of claim 1, wherein a SHA-256 hash value is calculated.

5. The method of claim 1, wherein the key is a AES-256 key.

6. The method of claim 1, wherein the encoding step uses Base64.

7. The method of claim 1, wherein a CPUID instruction from the processor is used to generate the first local configuration word.

8. The method of claim 1, wherein a 48-bit MAC address from a network interface card is used to generate the first local configuration word.

9. The method of claim 1, wherein a software license number from an operating system is used to generate the first local configuration word.

10. The method of claim 1, wherein a model and serial number from a peripheral device connected to the processor is used to generate the first local configuration word.

11. The method of claim 1, further comprising the processor performing the steps of:

decoding the encrypted string to get binary ciphertext;

obtaining the data word and calculating the hash value using the hash function to obtain a reconstructed key;

decrypting the binary ciphertext with the reconstructed key to obtain the password.

12. The method of claim 11comprising the processor using the password to access a computer system.

13. The method of claim 11, wherein the reconstructed key is stored in a memory by the processor and, after using the reconstructed key, the processor erases the reconstructed key from the memory.

14. The method of claim 13, wherein the processor erases the reconstructed key by writing over the reconstructed key in the memory.

15. A computer system having a plurality of associated local configuration words, comprising:

a memory having an instruction set stored in it;

a processor in communication with the instruction set in the memory, the instruction set operable to cause the processor to:

generate a data word from one of the local configuration words;

calculating a hash value from the data word;

generate an encrypted string by using the hash value as a key to encrypt the password; and

store the encrypted string in the memory.

16. The computer system of claim 15, wherein the instruction set causes the encrypted string in a software program which is stored the encrypted string in the memory.

17. The computer system of claim 15, wherein the data word is generated from one or more other of the local configuration words, the local configuration words being concatenated to generate the data word.

18. The computer system of claim 15 wherein the encrypted string is stored in a configuration file.

19. The computer system of claim 15 wherein the local configuration data word and the second local configuration data word are selected from the group consisting of: a string generated by using a CPUID instruction from the processor, a string generated by processing a 48-bit MAC address from a network interface card, a string generated by processing a software license number from an operating system, a string generated by processing a model and serial number from a peripheral device connected to the processor, and a string generated by processing model or serial numbers or interrupt vectors or addresses or priorities of the peripheral device.

20. The computer system of claim 15 wherein the processor, some time later, decodes the encrypted string to get binary ciphertext, obtains the local configuration word and the second local configuration word, concatenates the local configutation word and the second local configuration word, calculates the hash value to obtain a reconstructed key, decrypts the binary ciphertext with the reconstructed key to obtain the password, and uses the password to access a second computer.

21. A system, comprising:

a first computer having a memory with an instruction set and a processor in communication with the instruction set, and

a second computer that can be accessed with a password,

wherein the processor, under control of the instruction set, is operable to select a set of local configuration data, to concatenate the set of local configuration data and to calculate a hash value of the concatenated data, to generate an encrypted string by using the hash value as a key to encrypt the password and to encode the encrypted string as a string in a software program

wherein the processor, some time later under control of the instruction set, is operable to decode the encrypted string to get binary ciphertext, to obtain the set of local configuration data and calculates the hash value to obtain a reconstructed key, to decrypt the binary ciphertext with the reconstructed key to obtain the password, and to use the password to access the second computer.

AMENDED CLAIMS
received by the International Bureau on 06 January 2014 (06.01.2014)

1. A method of protecting a password in a computer, comprising a processor:

selecting from a plurality of system components in the computer a first local configuration word and combining the selected first local configuration words of the computer as a data word;

applying a hash function to the data word to calculate a hash value; and

generating an encrypted string by using the hash value as a key to encrypt the password, the password applied by the computer to automatically access a second computer;

storing the encrypted string in a memory in the computer; and

securely removing the key from the computer.

2. The method of claim 1, comprising the processor, encoding the encrypted string as a string in a software program which is stored in the memory.

4. The method of claim 1, wherein a SHA-256 hash value is calculated.

5. The method of claim 1, wherein the key is a AES-256 key.

6. The method of claim 1, wherein the encoding step uses Base64.

7. The method of claim 1, wherein a CPUID instruction from the processor is used to generate the first local configuration word.

8. The method of claim 1, wherein a 48-bit MAC address from a network interface card is used to generate the first local configuration word.

9. The method of claim 1, wherein a software license number from an operating system is used to generate the first local configuration word.

10. The method of claim 1, wherein a model and serial number from a peripheral device connected to the processor is used to generate the first local configuration word.

11. The method of claim 1, further comprising the processor performing the steps of:

decoding the encrypted string to get binary ciphertext;

obtaining the data word including combining identifying data from a plurality of system components in the computer and calculating the hash value from the obtained data word using the hash function to obtain a reconstructed key from the data word;

decrypting the binary ciphertext with the reconstructed key to obtain the password;

applying the obtained password to access the second computer.

15. A computer system having a plurality of associated local configuration words related to different system components, comprising:

a memory having an instruction set stored in it;

a processor in communication with the instruction set in the memory, the instruction set operable to cause the processor to:

generate a data word from the plurality of the local configuration words;

calculating a hash value from the data word;

generate an encrypted string by using the hash value as a key to encrypt the password, the password to be applied to access a second computer system;

store the encrypted string in the memory; and

securely removing the key from the computer.

18. The computer system of claim 15 wherein the encrypted string is stored in a configuration file.

19. The computer system of claim 15, wherein the plurality of local configuration words is selected from the group consisting of: a string generated by using a CPUID instruction from the processor, a string generated by processing a 48-bit MAC address from a network interface card, a string generated by processing a software license number from an operating system, a string generated by processing a model and serial number from a peripheral device connected to the processor, and a string generated by processing model or serial numbers or interrupt vectors or addresses or priorities of the peripheral device.

20. A system, comprising:

a first computer having a memory with an instruction set and a processor in communication with the instruction set, and

a second computer that can be automatically accessed with a password,

wherein the processor, under control of the instruction set, is operable to select a set of local configuration data from a plurality of system components, to concatenate the set of local configuration data and to calculate a hash value of the concatenated data, to generate an encrypted string by using the hash value as a key to encrypt the password and to encode the encrypted string as a string in a software program,

wherein the processor, some time later under control of the instruction set, is operable to decode the encrypted string to get binary ciphertext, to obtain the set of local configuration data

and calculates the hash value to obtain a reconstructed key, to decrypt the binary ciphertext with the reconstructed key to obtain the password, to use the password to automatically access the second computer; and securely removing the reconstructed key from the computer.

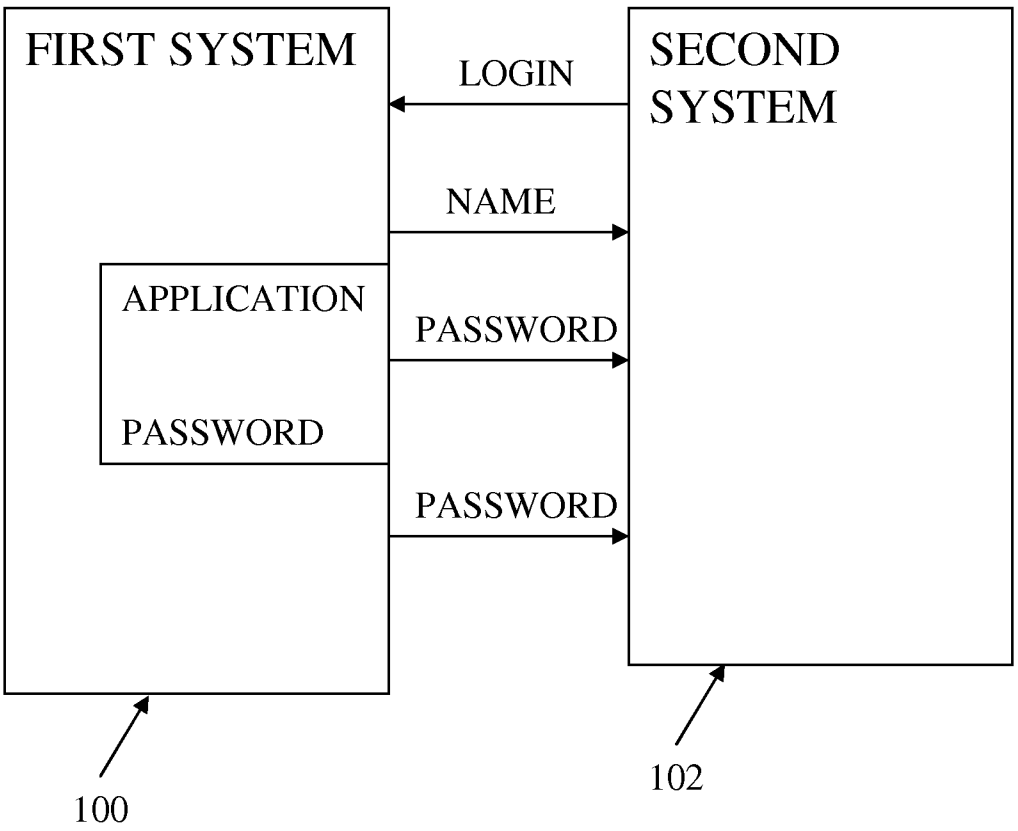
FIRST SYSTEM
LOGIN
SECOND SYSTEM
NAME
APPLICATION
PASSWORD
PASSWORD
PASSWORD
100
102

FIG. 1

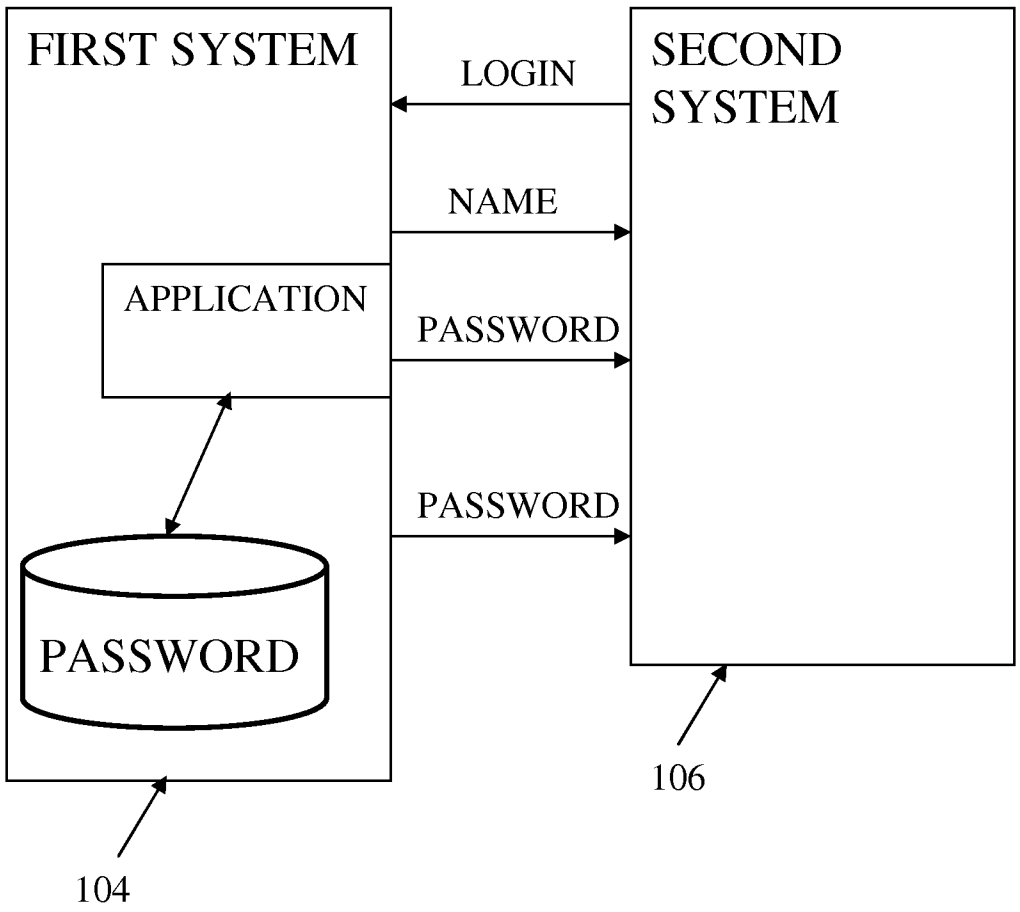
FIRST SYSTEM
LOGIN
SECOND SYSTEM
NAME
APPLICATION
PASSWORD
PASSWORD
PASSWORD
104
106

FIG. 2

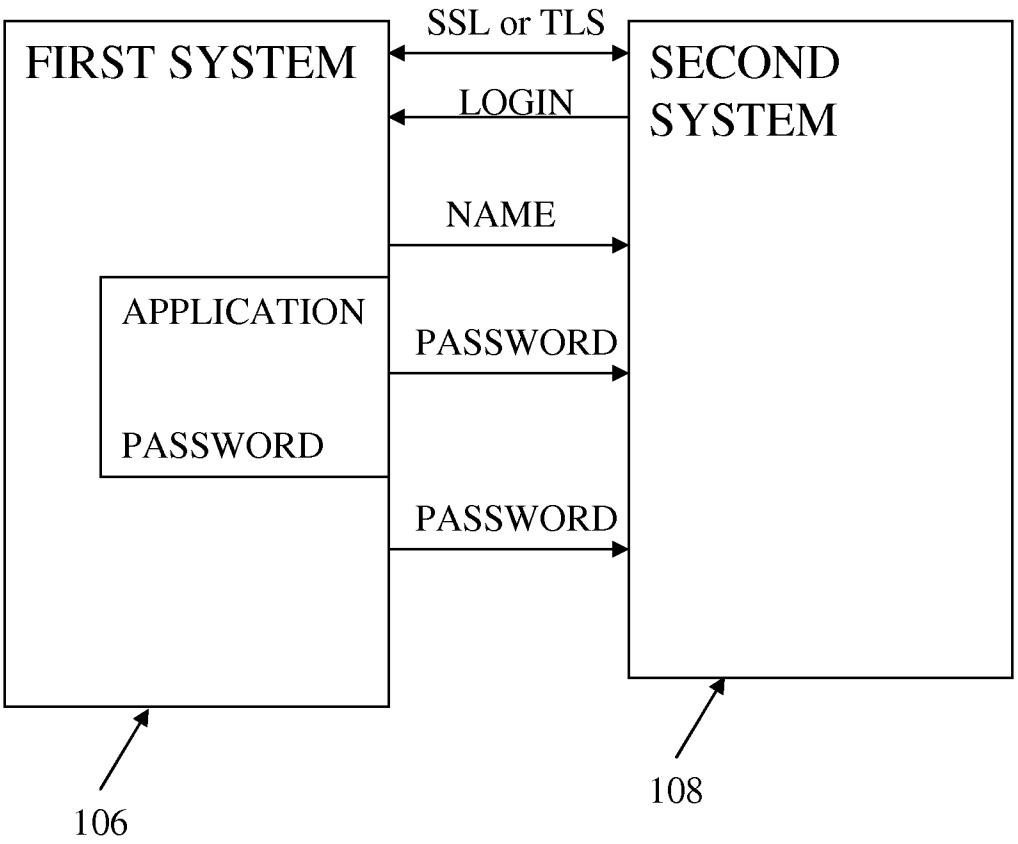
SSL or TLS
FIRST SYSTEM
SECOND SYSTEM
LOGIN
NAME
APPLICATION
PASSWORD
PASSWORD
PASSWORD
106
108

FIG. 3

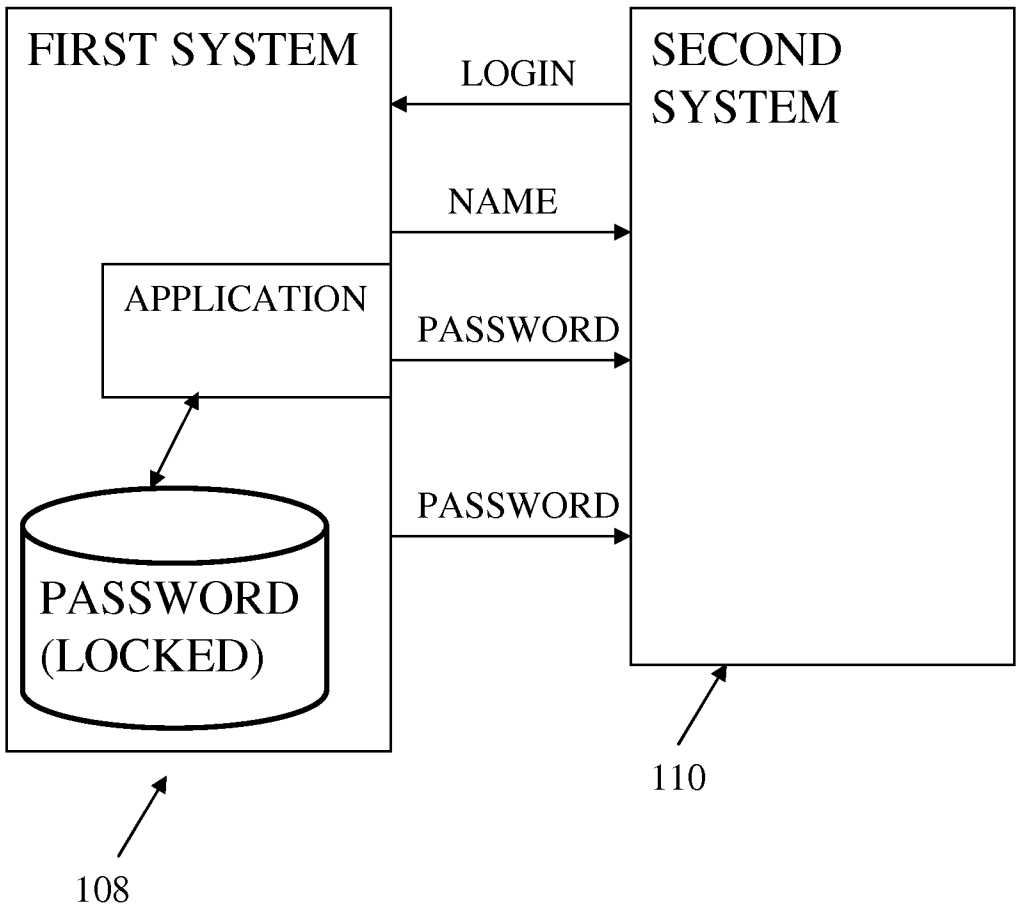
FIRST SYSTEM
LOGIN
SECOND SYSTEM
NAME
APPLICATION
PASSWORD
PASSWORD
PASSWORD (LOCKED)
110
108

FIG. 4

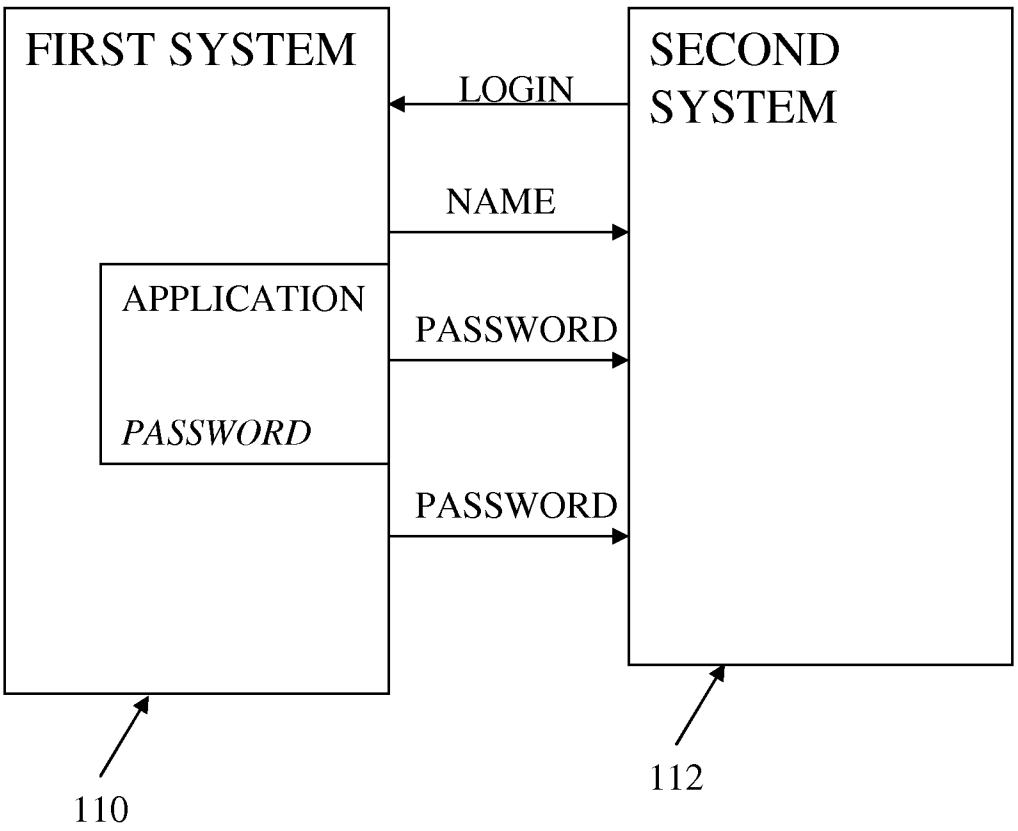
FIRST SYSTEM
SECOND SYSTEM
LOGIN
NAME
APPLICATION
PASSWORD
PASSWORD
PASSWORD
110
112

FIG. 5

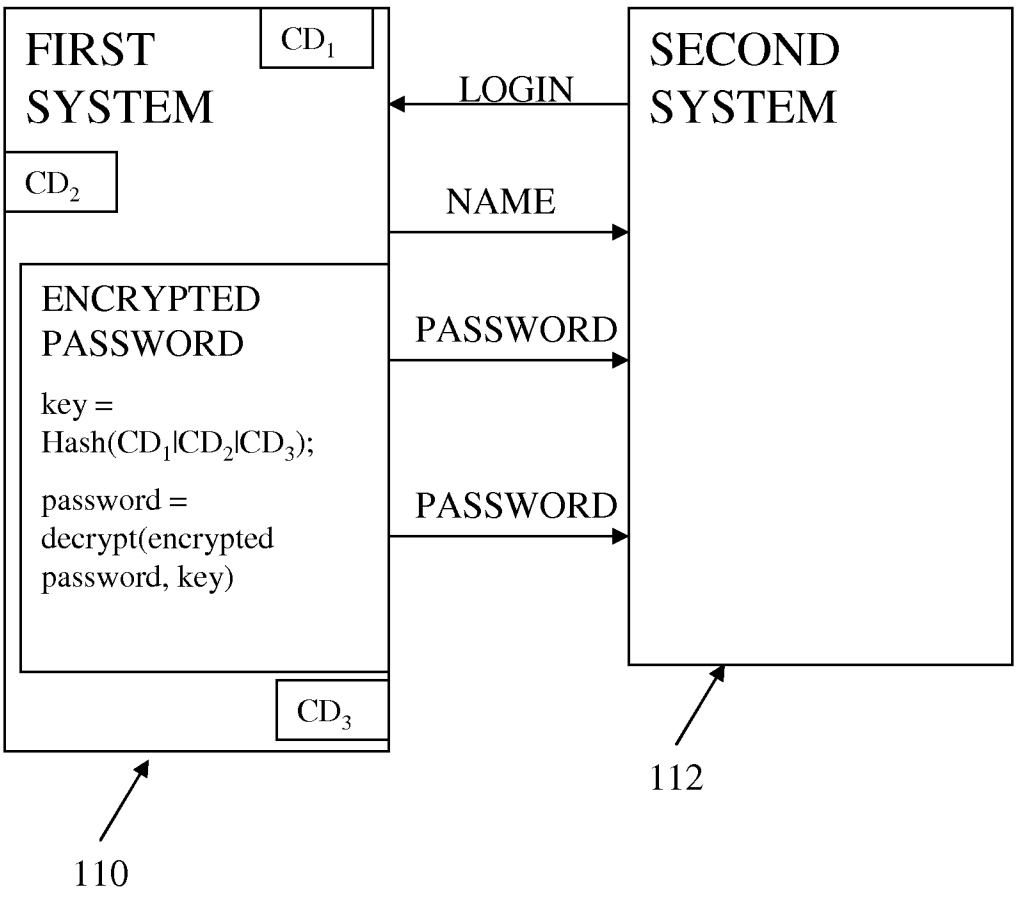
FIRST SYSTEM
CD1
CD2
ENCRYPTED PASSWORD
key = Hash(CD1|CD2|CD3);
password = decrypt(encrypted password, key)
CD3
110
LOGIN
NAME
PASSWORD
PASSWORD
SECOND SYSTEM
112

FIG. 6

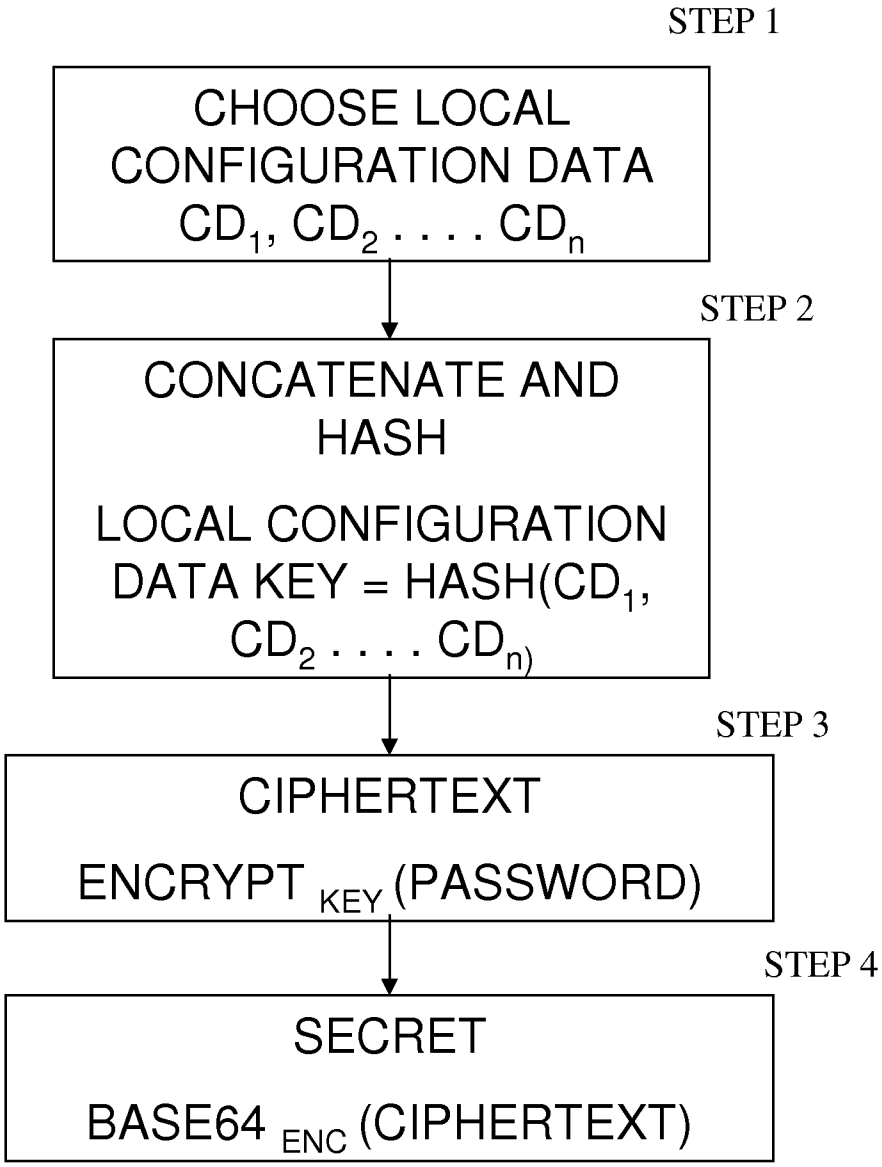
STEP 1
CHOOSE LOCAL CONFIGURATION DATA CD_1, CD_2 . . . . CD_n
STEP 2
CONCATENATE AND HASH
LOCAL CONFIGURATION DATA KEY = HASH(CD_1, CD_2 . . . . $CD_{n)}$
STEP 3
CIPHERTEXT
ENCRYPT $_{KEY}$ (PASSWORD)
STEP 4
SECRET
BASE64 $_{ENC}$ (CIPHERTEXT)

FIG. 7

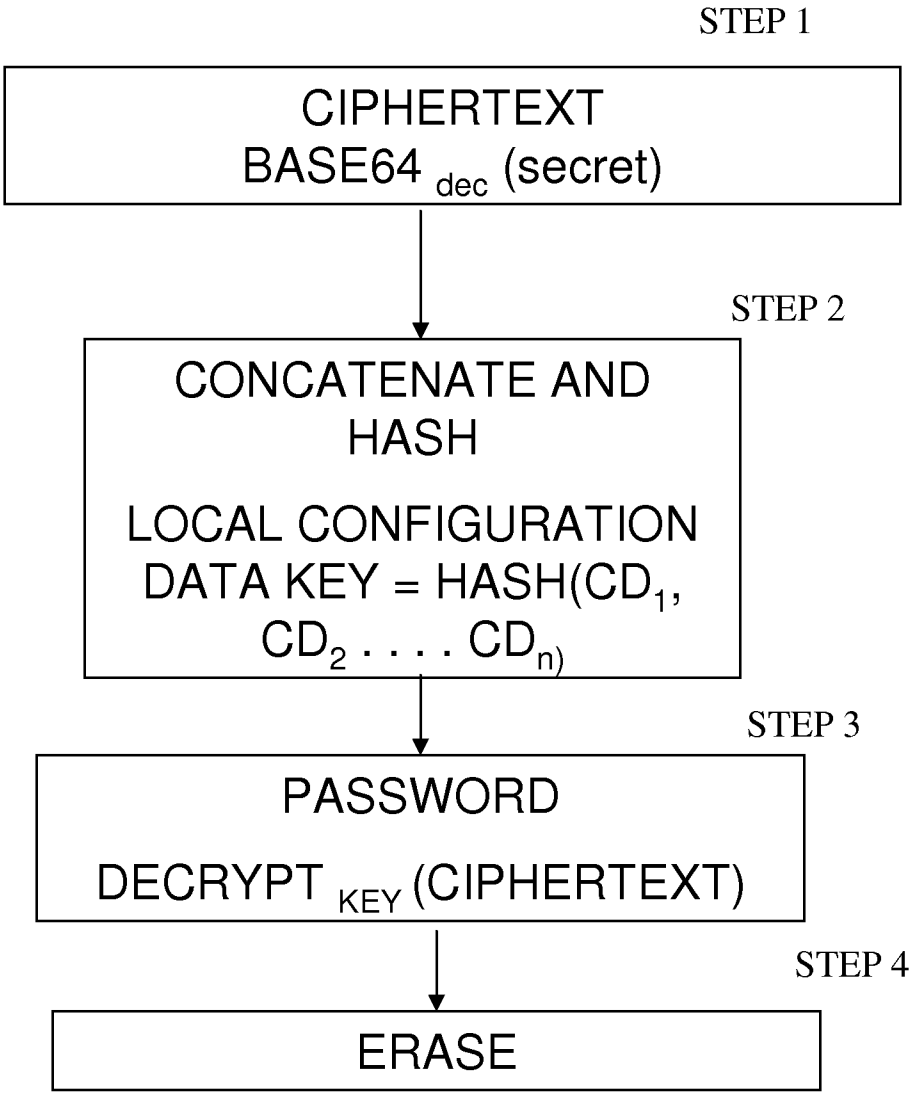
STEP 1
CIPHERTEXT
BASE64 dec (secret)
STEP 2
CONCATENATE AND HASH
LOCAL CONFIGURATION DATA KEY = HASH(CD1, CD2 . . . . CDn)
STEP 3
PASSWORD
DECRYPT KEY (CIPHERTEXT)
STEP 4
ERASE

FIG. 8

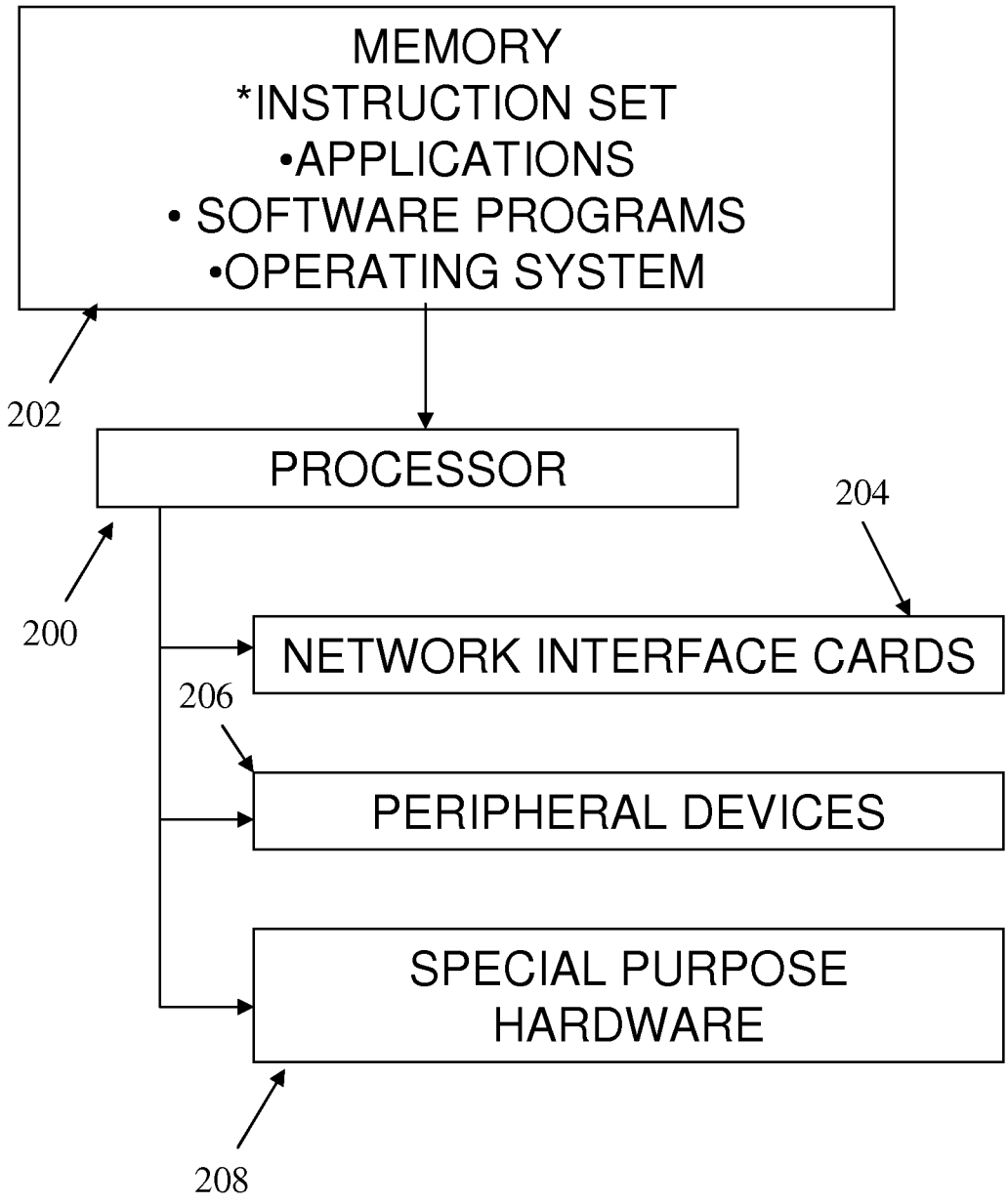
MEMORY
*INSTRUCTION SET
•APPLICATIONS
• SOFTWARE PROGRAMS
•OPERATING SYSTEM
202
PROCESSOR
204
200
NETWORK INTERFACE CARDS
206
PERIPHERAL DEVICES
SPECIAL PURPOSE HARDWARE
208

FIG. 9

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2013/052194

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/30 G06F21/62
ADD.

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)
G06F 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)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2006/156026 A1 (UTIN DANIIL [US]) 13 July 2006 (2006-07-13) paragraph [0005] paragraphs [0024] - [0047] -----	1-21
Y	US 2007/192631 A1 (ANDERSON DAVID B [US] ANDERSON DAVID BRUCE [US]) 16 August 2007 (2007-08-16) paragraph [0002] - paragraph [0003] paragraph [0016] - paragraph [0018]; figure 2 paragraph [0023]; figure 5 paragraph [0026] paragraph [0026] paragraph [0005] ----- -/--	1-21

[X] Further documents are listed in the continuation of Box C.

[X] 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
25 October 2013	07/11/2013

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Preuss, Norbert

2

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2004/123105 A1 (HIMMEL MARIA AZUA [US] ET AL) 24 June 2004 (2004-06-24) paragraph [0004] - paragraph [0005] paragraph [0077] - paragraph [0102]; claim 6 paragraph [0155] - paragraph [0186] -----	1-21

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2013/052194

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2006156026 A1	13-07-2006	AU 2003301719 A1	25-05-2004
		EP 1558983 A2	03-08-2005
		US 2006156026 A1	13-07-2006
		WO 2004040410 A2	13-05-2004
US 2007192631 A1	16-08-2007	JP 4646927 B2	09-03-2011
		JP 2007195190 A	02-08-2007
		US 2007192631 A1	16-08-2007
US 2004123105 A1	24-06-2004	NONE	