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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 November 2008 (27.11.2008)

PCT

(10) International Publication Number
WO 2008/141992 A1

(51) International Patent Classification:
G06F 21/22 (2006.01) *H04L 9/08* (2006.01)

(21) International Application Number:
PCT/EP2008/055912

(22) International Filing Date: 14 May 2008 (14.05.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
200710107636.6 23 May 2007 (23.05.2007) CN

(71) Applicant *(for all designated States except US)*: **SIEMENS AKTIENGESELLSCHAFT** [DE/DE]; Wittelsbacherplatz 2, 80333 München (DE).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **TANG, Wen** [CN/CN]; 7 Wangjing Zhonghuan Nanlu, Beijing 100102 (CN). **HU, Jian Jun** [CN/CN]; 7 Wangjing Zhonghuan Naniu, Beijing 100102 (CN).

(74) Common Representative: **SIEMENS AKTIENGESELLSCHAFT**; Postfach 22 16 34, 80506 München (DE).

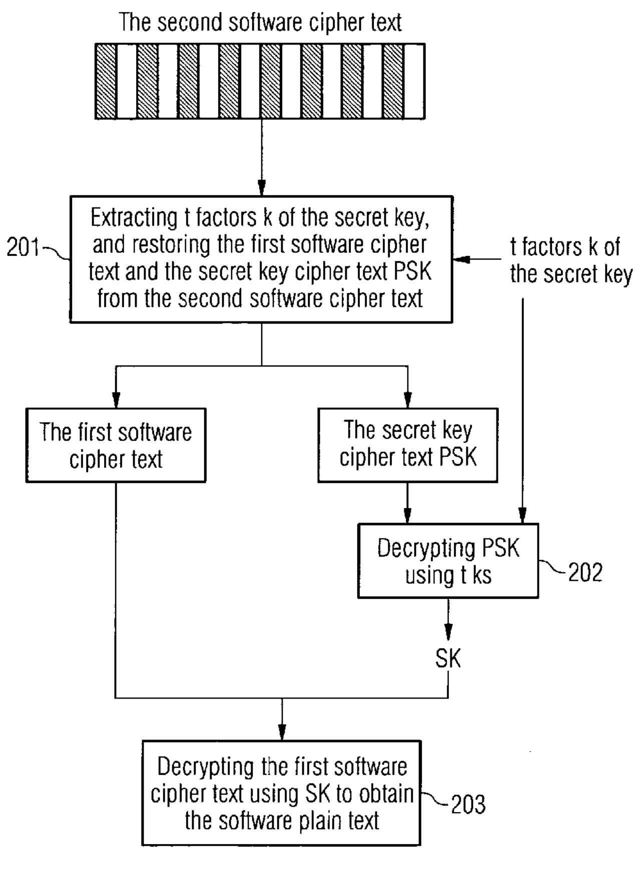
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, 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, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR ENCRYPTING AND DECRYPTING SOFTWARE

FIG 2


The second software cipher text
Extracting t factors k of the secret key, and restoring the first software cipher text and the secret key cipher text PSK from the second software cipher text
201
t factors k of the secret key
The first software cipher text
The secret key cipher text PSK
Decrypting PSK using t ks
202
SK
Decrypting the first software cipher text using SK to obtain the software plain text
203

(57) Abstract: The present invention relates to the field of computer security, and particularly to a method and an apparatus for encrypting and decrypting software. The decryption process of the present invention comprises the following steps: step 201, selecting *t* factors of a threshold secret key from *n* paragraphs of a second software cipher text at random, restoring a first software cipher text and an secret key cipher text PSK from the second software cipher text, wherein *n* is a positive integer greater than 1, *t* is a positive integer less than or equal to *n*; step 202, extracting said secret key cipher text PSK, calculating a second secret key according to said *t* factors of the threshold secret key, and using the second secret key to decrypt the secret key cipher text PSK into the first secret key SK; and step 203, decrypting said first software cipher text using said first secret key SK, so as to obtain the software's plaintext. The beneficial effects of the present invention are that it enhances the protection of the software encrypting key, and makes it more difficult for a cracker to crack the software by way of tracking the software' s loading process.

WO 2008/141992 A1

ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

Description

Method and apparatus for encrypting and decrypting software

Technical field

The present invention relates to the field of computer security, in particular to the field of computer encryption, and specifically to a method and an apparatus for encrypting and decrypting software.

Background art

Nowadays, software has become a commodity with independent value, and the functions, executing processes, coding, etc. in a piece of software tend to become objects plagiarized by competitors and other organizations or individuals. Therefore, software, particularly the software programmed in an intermediate language such as an programming language like Java, .NET, and so on are very easy to be encoded reversely by reverse engineering, for example by using .NET Reflect (a reverse engineering tool from Microsoft), JAD (a reverse engineering tool from Java), thereby obtaining the information regarding core algorithms, encoding and so on, and if such information is used in bad faith by crackers, for example by imitating the core algorithms of the software to bypass the registered software and so on, they will causes losses to the developers.

In the prior art, the measures for confusing the crackers in their cracking activities by changing the names of internal functions, rearranging the control flows or other methods have certain effects, which make the software programs reverse-coded difficult to read and understand or even impossible to read and understand, however this kind of protection mechanism to source codes cannot avoid the software program being encoded reversely, so it is still possible for the information of the software program to be lost.

In the article entitled "Using DES Encryption Algorithm to Protect Java Source Code", published in May 2005 in *Computer and information technology*, is disclosed a solution for encrypting software compiled by Java and decrypting it when running. The solution uses the Data Encryption Standard (DES) to encrypt a Java executable program, stores the encrypted program encoding and secret key in a memory, uses a loader to load the encrypted Java program encoding and key into system, takes out the secret key to decrypt the program encoding, converts it into the form of executable encoding, and loads it into a Java Virtual Machine to run.

The above method is very easy to be traced by crackers, and a cracker can trace every step from starting a program merely by using a debugging tool. If the program accesses a certain file every time when it is running, and obtains the secret key or the system's symbol name from the file, it will make the cracker to think that the file could be the secret key to the program or the comparison table for the system's symbol names, and if the cracker has confirmed that the file is the secret key file, then he will try every possible means to crack the file; and once the file is cracked, the software coding's cipher text can be converted into software coding's plaintext, and the source code of the software can be generated by reverse engineering, thereby causing a loss to the owner of the software.

Contents of the present invention

In order to solve the above problem and to enhance the degree of difficulty in software decompilation, an object of the present invention is to provide a method for encrypting software and a corresponding decryption method, wherein a threshold encryption feature is included, and every time when starting the software the address of threshold secret key factors obtained is different, which makes a cracker unable to decide which one is the secret key address.

The present invention also provides an apparatus for encrypting software and a corresponding decryption apparatus, which can store a plurality of factors of a threshold secret key into different paragraphs of the software and at the time of decryption it can obtain the factors of the threshold secret key from some paragraphs at random for decrypting the software.

Step 101: encrypting a software plaintext in a storage medium into a first software cipher text by using a first encryption module, wherein a secret key for decryption is a first secret key SK;

step 102: generating a second secret key by a second encryption module using *n* factors of a threshold secret key, encrypting said first secret key SK into an secret key cipher text PSK by using the second secret key, and splicing said secret key cipher text PSK into said first software cipher text, wherein *n* is a positive integer greater than 1; and

step 103: dividing said secret key cipher text PSK and said first software cipher text as an integrated whole into *n* paragraphs by using an encapsulation module, and splicing said factors of the threshold secret key into said paragraphs to form a second software cipher text which is stored in said storage medium.

According to a further aspect of the encryption method of the present invention, said encryption method specified in said step 101 comprises a symmetric encryption algorithm or an asymmetric encryption algorithm.

According to yet a further aspect of the encryption method of the present invention, the threshold secret key algorithm used in said step 102 comprises a Shamir threshold secret key scheme.

According to another further aspect of the encryption method of the present invention, in said step 103, said encapsulation module divides said secret key cipher text PSK and the first software cipher text as the integrated whole into *n* paragraphs; *C* represents any paragraph in said *n* paragraphs, and the paragraph *C* comprises blocks C_0, C_2, …, C_{m-1}, and the following calculations are performed on each paragraph *C* and its corresponding *k*:

$$C'_0 = C_0 \times k \qquad \text{(E0)}$$

$$C'_1 = C_1 \times k + C_0 \qquad \text{(E2)}$$

$$C'_2 = C_2 \times k + C_1 \qquad \text{(E3)}$$

… … …

$$C'_{m-1} = C_{m-1} \times k + C_{m-2} \qquad (\mathrm{E}m-1)$$

$$C'_m = C_{m-1} \qquad (\mathrm{E}m)$$

in which × is the arithmetic multiplication operation, at the same time a hash value *h* of the threshold secret key factor *k* is calculated, the values of C'_0 to C'_m are combined to form C', and the C's of the *n* paragraphs and their corresponding hash values *h* are spliced together to form said second software cipher text.

A method for decrypting software, comprising the following steps during the process of loading the software:

step 201: selecting *t* factors of a threshold secret key by a decapsulation module from *n* paragraphs of a second software cipher text at random; and restoring a first software cipher text and an secret key cipher text PSK from said second software cipher text, wherein *t* is greater than or equal to 1 and less than or equal to *n*, and *n* is a positive integer greater than 1;

step 202: extracting said secret key cipher text PSK, generating a second secret key by a second decryption module according to said *t* factors of the threshold secret key, and decrypting the secret key cipher text PSK into a first secret key SK by using the second secret key; and

step 203: decrypting said first software cipher text by a first decryption module using said first secret key SK, and transmitting a software plaintext to a CPU, so as to execute the software.

According to a further aspect of the decryption method of the present invention, in said step 201, said decapsulation module performs calculation on each of the *n* paragraphs of the second software cipher text: eliminating C_0, C_1, … C_{m-1} according to E0 to E*m*, so as to obtain the equation

$$0 = -C'_m k^m + C'_{m-1} \times k^{m-1} - C'_{m-2} \times k^{m-2} + \ldots + (-1)^{m-1} \times C'_0 \quad \text{(P0)},$$

k in the equation is solved, when the hash value of *k* is equal to the corresponding hash value *h* of the paragraph C', the values of C_0 to C_{m-1} are restored from C'_0 to C'_{m-1} by using *k*, C_0 to C_{m-1} are combined to obtain the paragraph *C* which is one of the *n* paragraphs of the integrated whole of the first software cipher text and the secret key cipher text; *n* *k*s are solved, and the first software cipher text and the secret key cipher text PSK are restored from the second software cipher text.

According to yet a further aspect of the decryption method of the present invention, a polynomial Newton iteration method is used to solve *k* in said equation (P0).

An apparatus for encrypting software, characterized in that it comprises a first encryption module, a second encryption module and an encapsulation module; said first encryption module encrypts a software plaintext to a first software cipher text using a first secret key SK; said second encryption module, which is connected with said first encryption module, generates a second encryption module using *n* factors of a threshold secret key, encrypts said first secret key SK into an secret key cipher text PSK using the second secret key, and stores said secret key cipher text PSK into said first software cipher text; and said encapsulation module, which is connected with said second encryption module, divides said first software cipher text into *n* paragraphs, and splices said factors of the threshold secret key into said paragraphs to form a second software cipher text.

An apparatus for decrypting software, characterized in that it comprises a decapsulation module, a second decryption module and a first decryption module; said decapsulation module decapsulates a second software cipher text into a first software cipher text and an secret key cipher text PSK, and selects *t* factors of a threshold secret key from *n* paragraphs of the second software cipher text at random; said second decryption module, which is connected with said decapsulation module, generates a second secret key according to said *t* factors of the threshold secret key, and decrypts the secret key cipher text PSK into the first secret key SK by using the second secret key; said first decryption module, which is connected with said second decryption module, decrypts said first software cipher text by using said first secret key SK, obtains a software plaintext and transmits the same to a CPU so as to execute the software.

The beneficial effects of the present invention are that it enhances the protection of the software encrypting key, and makes it more difficult for a cracker to crack the software by way of tracking the software’s loading process, obtaining the physical address of the secret key by tracing the software loading process, thereby achieving the purpose of cracking the software by analyzing the secret key, and the present invention enhances the current solution of encrypting the software to improve the security thereof by the technology of dynamically storing the secret key.

Brief description of the drawings

Fig. 1 is a flowchart of performing the software encryption according to the present invention;

Fig. 2 is a flowchart of performing the software decryption according to the present invention;

Fig. 3 is a structure scheme of an apparatus for performing the software encryption according to the present invention;

Fig. 4 is a structure scheme of an apparatus for performing the software decryption according to the present invention; and

Fig. 5 is a structural diagram of an apparatus for implementing the present invention.

Detailed description of the preferred embodiments

Hereinbelow, the present invention is explained in detail in combination with the drawings.

The present invention utilizes the theory of a threshold secret key to provide further protection to said first secret key, and splices the factors of the threshold secret key into the encrypted software, so as to make a cracker obtain a different jump address every time he traces the program running, so that the cracker will not be able to determine where to seek said first secret key. The software that can be protected by the present invention is not only limited to executable programs, but also includes functional modules and the software's core algorithms and so on. The current threshold encryption method is to encrypt said first secret key SK to a secret key cipher text PSK by using a random number as a second secret key, and at the same time generates *n* factors of the threshold secret key for computing the random number; at the time that the secret key needs to be decrypted, it only needs *t* factors of the threshold secret key ($t \leq n$) to generate said second secret key for decryption. The purpose for proposing the threshold cryptography is to disperse the rights and to enhance the security; the dispersion of rights is demonstrated in that when using the threshold cryptography for performing the decryption and if every person holds one secret key factor, the decryption can be accomplished only if the number of participators reach a certain number (the threshold value *t*); security, on the one hand, is to prevent the case that obtaining one key factor makes the encryption meaningless, therefore as long as the number of cracked persons in this group does not reach the threshold value it is still impossible to do the decryption; on the other hand, it is to prevent the case of the loss of a key factor affecting the normal decryption, since the decryption can still be carried out as long as the number of persons having valid key factors is greater than or equal to the threshold value. In the embodiments of the present invention the threshold encryption algorithm uses the Shamir scheme as an example, but it is not limited to the Shamir scheme, it is also possible to use the Asmuth-Bloom threshold secret key scheme.

Before selling a piece of software, the vendor of the software encrypts the software plaintext by using an encryption algorithm which is the currently available symmetric or asymmetric encryption algorithm such as AES, DES or RSA, ECC and so on. If the symmetric encryption algorithm is used, then the encryption secret key of software is the same as the decryption secret key, which can also be used for decryption, and the decryption secret key is the secret key SK (namely, the first secret key). If the asymmetric encryption algorithm is used, then the encryption secret key has a corresponding relationship with the decryption secret key of said asymmetric encryption algorithm, and the decryption secret key is the secret key SK (namely, the first secret key) in the present invention. Since the software's secret key SK is critical to whether the software can be cracked or not, the security of the secret key SK is very important, and the present invention specifically uses the Shamir scheme of threshold encryption to generate a second secret key by calculating *n* factors of the threshold secret key, K_1, K_2, …, K_n, the second secret key is used to encrypt the secret key SK into a secret key cipher text PSK, and the secret key cipher text PSK is spliced into the encrypted software, for example it is spliced into the head or tail of the encrypted software. Furthermore, the *n* factors of the threshold secret key are spliced into different physical paragraphs of the encrypted software by a strong splicing algorithm (or a simple splicing mode), for example, they splice it into the head or tail of the software. In the present invention, it is carried out by: step one, encrypting the software which needs to be protected; step two, encrypting the first secret key SK from step one; and step three, splicing the secret key factors for performing the encryption in step two; while when the software needs to be decrypted in order to run, t ($1 \leq t \leq n$, both t and n are positive integers) factors of the threshold secret key are obtained from the protected software cipher text at random, and then the first secret key SK of encrypted software can be solved from the secret key cipher text PSK by using the Shamir scheme, so as to decrypt the software cipher text. The restoration method of the threshold secret key makes the software loading process have dynamic characteristics, and every time the threshold secret key factors for decryption are obtained from different positions of the software, so it can effectively increase the degree of difficulty for cracking by a cracking method of tracing the software loading process.

A flowchart of software encryption process of the present invention is shown in Fig. 1.

Step 101, select a suitable symmetric encryption algorithm such as AES, DES and so on, and encrypt the software plaintext into a first software cipher text by using a first encryption module, wherein the secret key used is a first secret key SK.

Step 102, protect the aforementioned secret key SK by using the Shamir algorithm in threshold encryption algorithms by a second encryption module, and use the Shamir scheme of Lagrange interpolation polynomial algorithm in a Z_P field to generate a t-1 order polynomial, where Z_P is a prime field:

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1}$$

wherein the coefficients a_0, … a_n of $P_n(x)$ are generated at random.

Let $x_1 = 1$, calculate $P_n(1) = a_0 + a_1 + a_2 + \ldots + a_{t-1}$,

… … …

let $x_n = n$, calculate $P_n(n) = a_0 + a_1 n + a_2 n^2 + \ldots + a_{t-1} n^{t-1}$.

wherein, $P_n(1),\ldots,P_n(n) < 2^{64}$, n is a positive integer greater than 1, and t is a positive integer greater than and equal to 1 and less than n.

Then generate n factor pairs of the threshold secret key $K_1 = (1, P_n(1))$, … $K_n = (n, P_n(n))$, encrypt the secret key SK into the secret key cipher text PSK using a_0 as the second secret key. Also, splice the secret key cipher text PSK after having been encrypted into the head or tail of said first software cipher text, and in this step a storage method in the prior art can be used.

Step 103, divide the integrated whole of the first software cipher text and key cipher text into n paragraphs by using the encapsulation module, and splice the n factors of the threshold secret key into *n* paragraphs. Here, the *n* factors of the secret key can be directly spliced respectively into the head and tail of each paragraph of the first software cipher text, so as to form the second software cipher text which is stored in storage medium, as shown in this figure, the black parts are the factors of the secret key and the white parts are the *n* paragraphs; and it is also possible to use the following splicing method to form a more complicated second software cipher text.

C represents a certain paragraph of the first software cipher text, wherein each paragraph *C* comprises C_0, C_2, ..., C_{m-1}, *k* represents $P_n(i)$ of threshold secret key factor pair K_i, and the specific splicing process is as follows:

$$C_0' = C_0 \times k \qquad \text{(E0)}$$

$$C_1' = C_1 \times k + C_0 \qquad \text{(E2)}$$

$$C_2' = C_2 \times k + C_1 \qquad \text{(E3)}$$

$$\cdots \quad \cdots \quad \cdots$$

$$C_{m-1}' = C_{m-1} \times k + C_{m-2} \qquad (\mathrm{E}m-1)$$

$$C_m' = C_{m-1} \qquad (\mathrm{E}m)$$

wherein × is an arithmetic multiplication operation. As a preferred embodiment, the length of each paragraph C_i is equal to the length of *k*, that is to say length(C_i) = length(*k*). For example, the software is divided into *n* paragraphs after being encrypted, in which a certain paragraph *C* has a length of 128 bytes, while a factor of the secret key has a length of 16 bytes, then *C* is divided into 8 paragraphs, namely m = 7, and the length of each paragraph C_i in *C* has a length of 16 bytes. At the same time, h = hash(*k*) is calculated, namely the hash value of the threshold secret key factor *k* is recorded, which is used for verifying whether the restored threshold factor is correct or not at the time of decryption. After having combined C'_0 to C'_m to form a complete C', it is then spliced with the hash value h (h is added in front or behind the paragraph C'), and then the final software cipher text for storing, namely the second software cipher text, is formed by splicing all the paragraphs C's and the corresponding hash values h, and the second software cipher text is stored into the storage medium.

In the Shamir threshold encryption scheme, any t factors of the secret key can be used to restore the second secret key a_0, so as to decrypt PSK, and therefore the software loader will select t of n factors of the secret key at random for decrypting PSK every time when the encrypted software is loaded, so as to provide a highly powerful protection mechanism with dynamic characteristics to prevent a cracker from tracing and analyzing the software loading process.

Fig. 2 is a flowchart of loading and decrypting software according to the present invention. At the stage of starting the software, the second software cipher text is loaded into the memory from a storage medium by a loader, and in this figure the black parts are the secret key factors, and the white parts are the first software cipher text and PSK; if the splicing method as shown in step 103 is not used in the encryption step, and only n factors of the secret key are directly spliced into the head or tail of the corresponding paragraphs of the software cipher text, then t factors of the secret key can be obtained by directly selecting from the cipher text t paragraphs at random by the decapsulation module in step 201, and the second cipher text is restored into the first cipher text and PSK. If the splicing method as shown in step 103 is used during the encryption, then one cipher text paragraph C' and its corresponding hash value h are selected by the decapsulation module, and the factor k of the threshold secret key carried in that paragraph of the cipher text is restored. The restoration algorithm is as follows:

eliminating the C_0 to C_{m-1} from E0 to Em, substituting $C_{m-1} = C'_m$ into (Em-1) to obtain the equation $C_{m-2} = C'_{m-1} - C'_m \times k + C_m \times k^2$, which is substituted into (Em-2), ..., till (E0); and finally forming the polynomial $0 = -C'_m k^m + C'_{m-1} \times k^{m-1} - C'_{m-2} \times k^{m-2} + \ldots + (-1)^{m-1} \times C'_0$, which is labeled as P0; the secret key factors k are the roots of the aforementioned polynomial, and the roots are found by calculating the polynomial in the numerical field, and k can be restored from the second software cipher texts $C'_0, C'_1, \ldots, C'_m$. The Newton iteration algorithm is used to seek one or more roots of the polynomial P0 in this embodiment.

(a) Let $y = -C'_m k^m + C'_{m-1} \times k^{m-1} - C'_{m-2} \times k^{m-2} + \ldots + (-1)^{m-1} \times C'_0 = f(k) \quad \text{(P1)}$, arbitrarily selecting an initial k_0, for example $k_0 = 2^{lengh(k)-1}$.

(b) Calculate $k_{i+1} = k_i - \frac{f(k_i)}{f'(k_i)}$, i=0 to m , $f'(k)$ is the derivative of $f(k)$, namely,

$$f'(k) = -C'_m \times m \times k^{m-1} + C'_{m-1} \times (m-1) \times k^{m-2} - C'_{m-2} \times (m-2) \times k^{m-3} + \ldots + (-1)^{m-2} \times C'_1 .$$

(c) Repeat step b, till $|k_{i-1} - k_i| < 1$, then k_{i+1} is approximate to the root of P1.

(d) If hash(k_{i+1}) = h, or hash(k_{i+1}+1) = h, hash(k_{i+1}-1) = h, wherein h is the h value in the encryption step (4), then the k_{i+1} calculated in this step is the factor k of the threshold secret key in the encryption step, and jump to step (f); if it is not equal, then the algorithm for finding the numerical root k fails, enter into step (e). Said Hash algorithm in the present invention is a one-way algorithm, that is to say, the original data cannot be deduced inversely after the data are calculated, and therefore if it is necessary to compare whether the data are altered before and after they are transmitted, it is only necessary to make a comparison of the hash values before and after the transmission.

(e) If k is not found in step (d), then it means that P0 has several real roots, and the other real roots can be obtained by the following method:

Use the root k_{i+1} obtained in step (d) as a new k_0.

Let $b_0 = -C'_m$, $b_k = (-1)^{k-1} \times C'_{m-k} + k_0 \times b_{k-1}$, where k = 1, 2, …, m-1, and then establish a new polynomial, $f(k) = b_0 \times k^{m-1} + b_1 \times k^{m-2} + \ldots + b_{m-1}$ (P2) ;

use the aforementioned steps b-c to calculate the real roots of the new equation P2, so as to obtain the other real roots of P0.

Calculate all of the real roots of P0 by this step (e), and repeat to check step (d) every time after passing step (e) to determine whether or not the real secret key factor has been obtained, and then obtain the factor k of the threshold secret key.

(f) After having obtained the factor k of the secret key in a cipher text paragraph, substitute it back to E0 to Em, and restore the first software cipher text $C_1, C_2, \ldots, C_m$ from the second software cipher text $C'_0, C'_1, \ldots, C'_m$.

Perform steps a-f on n paragraphs C' to obtain all the factors k of the threshold secret key needed in decrypting the secret key cipher text PSK, and restore all the cipher text C' by using k to form the first software cipher text.

Step 202, after restoring the t factors of the threshold secret key,
K_i = $(x_i, P_n(x_i))$, $1 \le i \le t$, establish a new polynomial by using t

ks by the second decryption module

$$P_n(x) = \sum_{i=1}^{t} \left(\prod_{\substack{i=1 \\ i \neq k}}^{t} \frac{x - x_i}{x_k - x_i} \right) y_k$$

wherein, $y_k = P_n(x_k)$, x_i and x_k are x_i in the restored threshold secret key factor pair, in which i ≠ k, and finally let x = 0, to obtain $P_n(0) = a_0$.

Extract PSK in the first software cipher text, and use a_0 as the secret key for decrypting the secret key cipher text PSK, so as to obtain the first secret key SK for decrypting the encrypted software.

Step 203, decrypt the encrypted software by using SK by the first encryption module, so as to obtain the original software plaintext.

A CPU operates according to the software plaintext.

A schematic diagram of an encryption apparatus of the present invention is shown in Fig. 3, which comprises a first encryption module, a second encryption module and an encapsulation module; said first encryption module encrypts a software plaintext into a first software cipher text by using a first secret key SK; said second encryption module, which is connected with said first encryption module, generates a second encryption module using *n* factors of a threshold secret key, encrypts said first secret key SK into an secret key cipher text PSK by using the second secret key, and stores said secret key cipher text PSK into said first software cipher text; and said encapsulation module, which is connected with said second encryption module, divides said first software cipher text into *n* paragraphs, and splices said factors of the threshold secret key into said paragraphs to form a second software cipher text.

A schematic diagram of a decryption apparatus of the present invention is shown in Fig. 4, which comprises a decapsulation module, a second decryption module and a first decryption module; said decapsulation module decapsulates a second software cipher text into a first software cipher text, and selects *t* factors of a threshold secret key from *n* paragraphs of the second software cipher text at random; said second decryption module, which is connected with said decapsulation module, generates a second secret key according to said *t* factors of the threshold secret key, and decrypts the secret key cipher text PSK into the first secret key SK by using the second secret key; said first decryption module, which is connected with said second decryption module, decrypts said first software cipher text by using said first secret key SK, so as to obtain a software plaintext.

A schematic operation diagram of an apparatus of the present invention is shown in Fig. 5. It comprises a loader for loading software from a storage medium, and also the encryption apparatus as shown in Fig. 4, here redundant description on the same parts is not repeated. The loader loads the second software cipher text from the storage medium (for example a hard disk), and inputs it into said decryption apparatus which transforms said second software cipher text to the software plaintext, and then transmits it to the CPU for executing.

The beneficial effects of the present invention are that, it encrypts executable software so as to make it impossible for a cracker to obtain the secret key by simply tracing the software loading process, so that it prevents the software from being decrypted and compiled by way of reverse engineering and so on. It enhances the protection to the software’s secret key, and makes it more difficult for crackers to obtain the physical address of the secret key by tracing the software loading process so as to achieve the object of cracking the software by analyzing the secret key; and the present invention, by way of the technology of dynamically storing the secret key, enhances the currently available solutions of encrypting the software for improving the security thereof.

The above particular embodiments are only used to describe the present invention, not to define the present invention.

Claims

1. A method for encrypting software, comprising the following steps:

step 101: encrypting a software plaintext in a storage medium into a first software cipher text by using a first encryption module, wherein an secret key for decryption is a first secret key SK;

step 102: generating a second secret key by a second encryption module using *n* factors of a threshold secret key, wherein *n* is a positive integer greater than 1, encrypting said first secret key SK into an secret key cipher text PSK by using the second secret key, and splicing said secret key cipher text PSK into said first software cipher text; and

step 103: dividing said secret key cipher text PSK and said first software cipher text as an integrated whole into *n* paragraphs by using an encapsulation module, and splicing said factors of the threshold secret key into said paragraphs to form a second software cipher text which is stored in said storage medium.

2. The method for encrypting software as claimed in claim 1, characterized in that said encryption method specified in said step 101 comprises a symmetric encryption algorithm or an asymmetric encryption algorithm.

3. The method for encrypting software as claimed in claim 1, characterized in that the threshold secret key algorithm used in said step 102 comprises a Shamir threshold secret key scheme, or an Asmuth-Bloom threshold secret key scheme.

4. The method for encrypting software as claimed in claim 3, characterized in that, in said step 103, said encapsulation module divides said secret key cipher text PSK and the first software cipher text as the integrated whole into *n* paragraphs;

C represents any paragraph in said *n* paragraphs, and the paragraph *C* comprises blocks C_0, C_2, …, C_{m-1}, wherein the length of each paragraph is equal to the length of a threshold secret key factor *k*, and the following calculations are performed on each paragraph *C* and its corresponding *k*:

$$C'_0 = C_0 \times k \qquad \text{(E0)}$$

$$C'_1 = C_1 \times k + C_0 \qquad \text{(E2)}$$

$$C'_2 = C_2 \times k + C_1 \qquad \text{(E3)}$$

… … …

$$C'_{m-1} = C_{m-1} \times k + C_{m-2} \qquad \text{(E}m\text{-1)}$$

$$C'_m = C_{m-1} \qquad \text{(E}m\text{)}$$

in which × is the arithmetic multiplication operation, at the same time a hash value *h* of the threshold secret key factor *k* is calculated, the values of C'_0 to C'_m are combined to form C', and the C' s of the *n* paragraphs and their corresponding hash values *h* are spliced together to form said second software cipher text.

5. A method for decrypting software, comprising the following steps during the process of loading the software:

step 201: selecting *t* factors of a threshold secret key by a decapsulation module from *n* paragraphs of a second software cipher text at random, wherein *t* is greater than or equal to 1 and less than or equal to *n*, and *n* is a positive integer greater than 1; and restoring a first software cipher text and an secret key cipher text PSK from said second software cipher text;

step 202: extracting said secret key cipher text PSK, generating a second secret key by a second decryption module according to said *t* factors of the threshold secret key, and decrypting the secret key cipher text PSK into a first secret key SK by using the second secret key; and step 203: decrypting said first software cipher text by a first decryption module using said first secret key SK, and transmitting a software plaintext to a CPU, so as to execute the software.

6. The method for decrypting software as claimed in claim 5, characterized in that, in said step 201, said decapsulation module performs calculation on each of the *n* paragraphs of the second software cipher text: eliminating C_0, C_1, …, C_{m-1} according to E0 to E*m*, so as to obtain the equation

$$0 = -C'_m k^m + C'_{m-1} \times k^{m-1} - C'_{m-2} \times k^{m-2} + \ldots + (-1)^{m-1} \times C'_0 \quad \text{(P0)},$$

k in the equation is solved, when the hash value of *k* is equal to the corresponding hash value *h* of the paragraph C', the values of C_0 to C_{m-1} are restored from C'_0 to C'_{m-1} by using *k*, C_0 to C_{m-1} are combined to obtain the paragraph *C* which is one of the *n* paragraphs of the integrated whole of the first software cipher text and the secret key cipher text; *n* *k*s are solved, and the first software cipher text and the secret key cipher text PSK are restored from the second software cipher text.

7. The method for decrypting software as claimed in claim 6, characterized in that the polynomial Newton iteration method is used to solve *k* in said equation (P0).

8. An apparatus for encrypting software, characterized in that it comprises a first encryption module, a second encryption module and an encapsulation module; said first encryption module encrypts a software plaintext to a first software cipher text using a first secret key SK; said second encryption module, which is connected with said first encryption module, generates a second encryption module using *n* factors of a threshold secret key, encrypts said first secret key SK into an secret key cipher text PSK using the second

secret key, and stores said secret key cipher text PSK into said first software cipher text; and said encapsulation module, which is connected with said second encryption module, divides said first software cipher text into *n* paragraphs, and splices said factors of the threshold secret key into said paragraphs to form a second software cipher text.

9. An apparatus for decrypting software, characterized in that it comprises a decapsulation module, a second decryption module and a first decryption module; said decapsulation module decapsulates a second software cipher text into a first software cipher text and an secret key cipher text PSK, and selects *t* factors of a threshold secret key from *n* paragraphs of the second software cipher text at random; said second decryption module, which is connected with said decapsulation module, generates a second secret key according to said *t* factors of the threshold secret key, and decrypts the secret key cipher text PSK into the first secret key SK by using the second secret key; said first decryption module, which is connected with said second decryption module, decrypts said first software cipher text by using said first secret key SK, obtains a software plaintext and transmits the same to a CPU so as to execute the software.

FIG 1

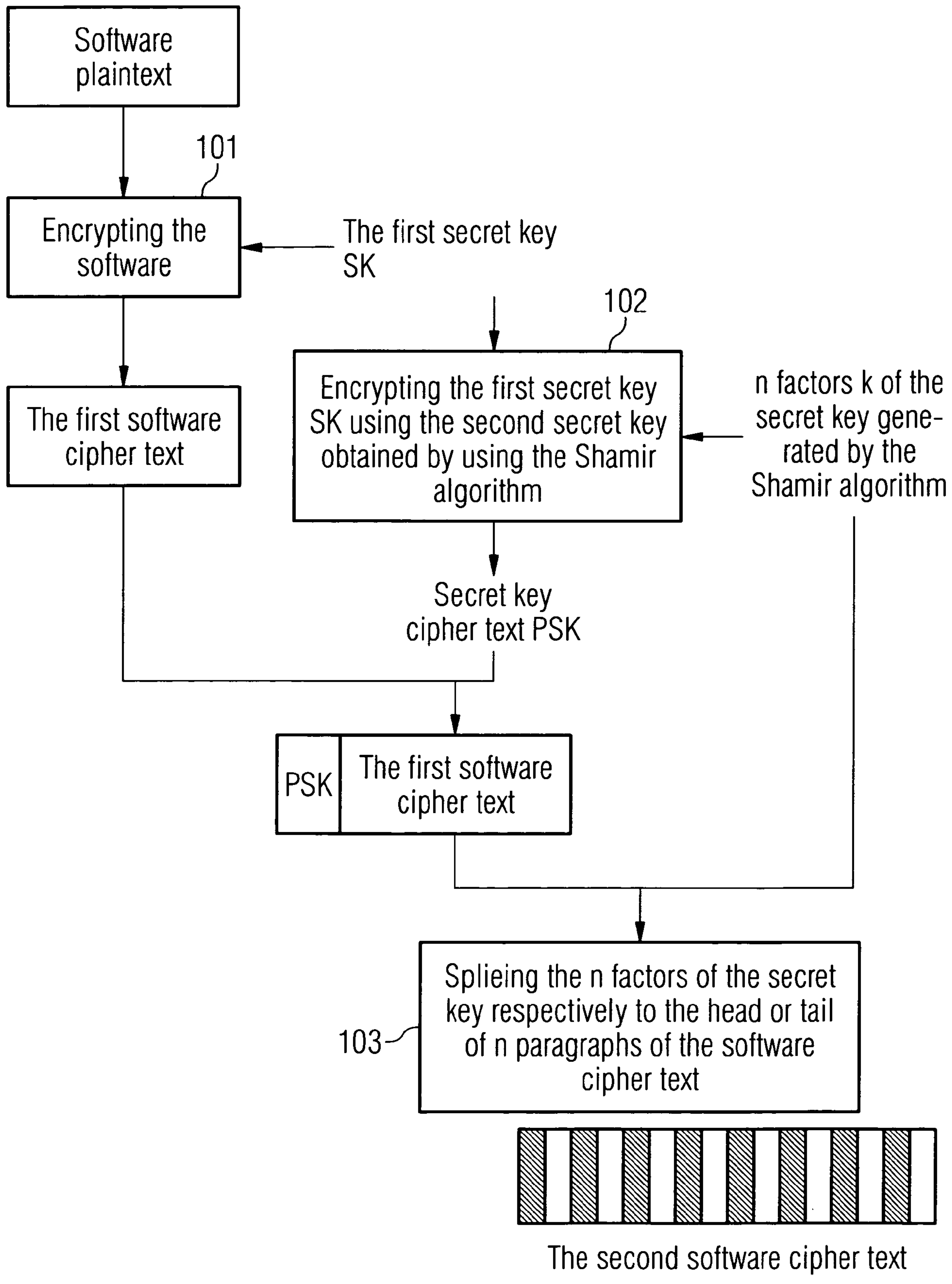

Software plaintext
101
Encrypting the software
The first secret key SK
The first software cipher text
102
Encrypting the first secret key SK using the second secret key obtained by using the Shamir algorithm
n factors k of the secret key generated by the Shamir algorithm
Secret key cipher text PSK
PSK
The first software cipher text
103
Splieing the n factors of the secret key respectively to the head or tail of n paragraphs of the software cipher text
The second software cipher text

FIG 2

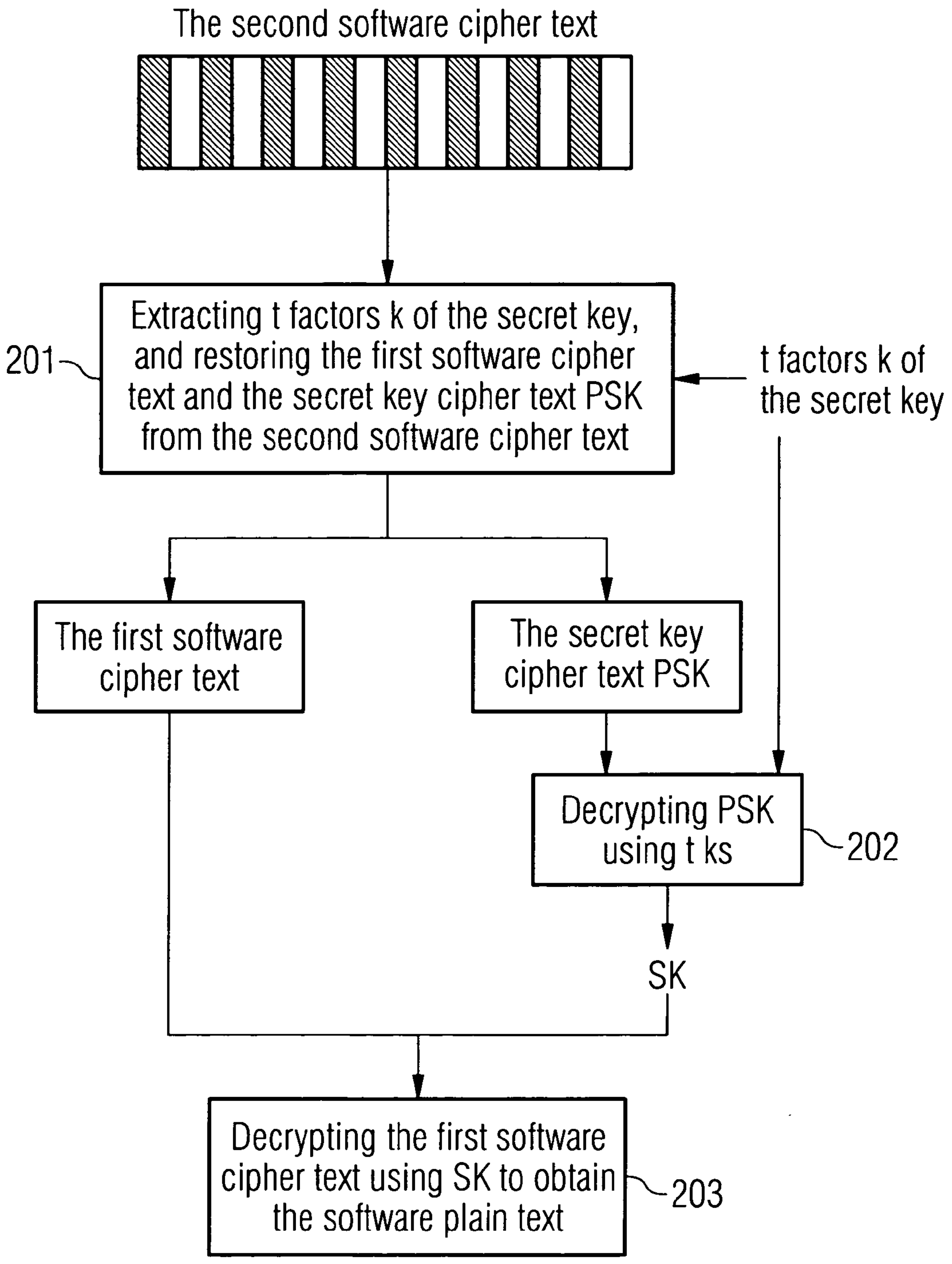

The second software cipher text
Extracting t factors k of the secret key, and restoring the first software cipher text and the secret key cipher text PSK from the second software cipher text
201
t factors k of the secret key
The first software cipher text
The secret key cipher text PSK
Decrypting PSK using t ks
202
SK
Decrypting the first software cipher text using SK to obtain the software plain text
203

FIG 3

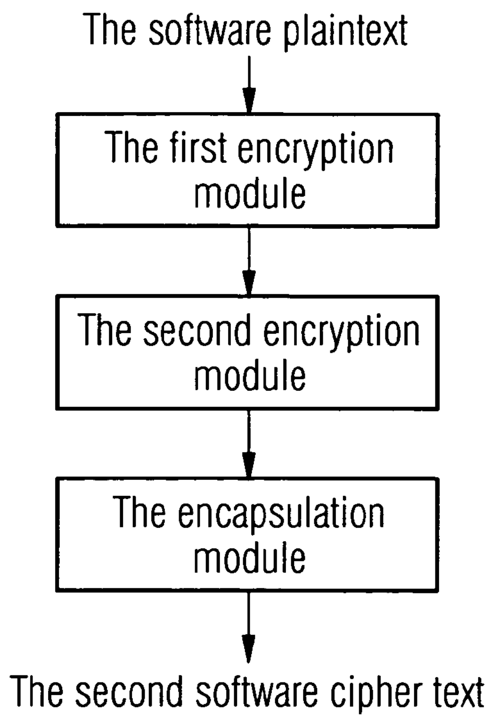

The software plaintext
The first encryption module
The second encryption module
The encapsulation module
The second software cipher text

FIG 4

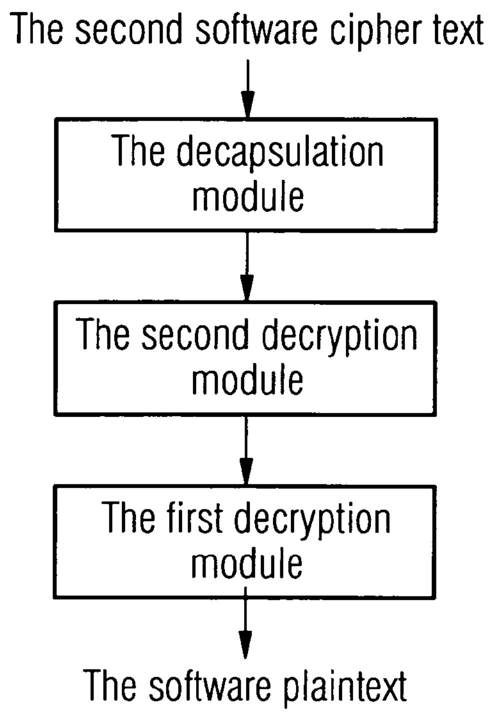
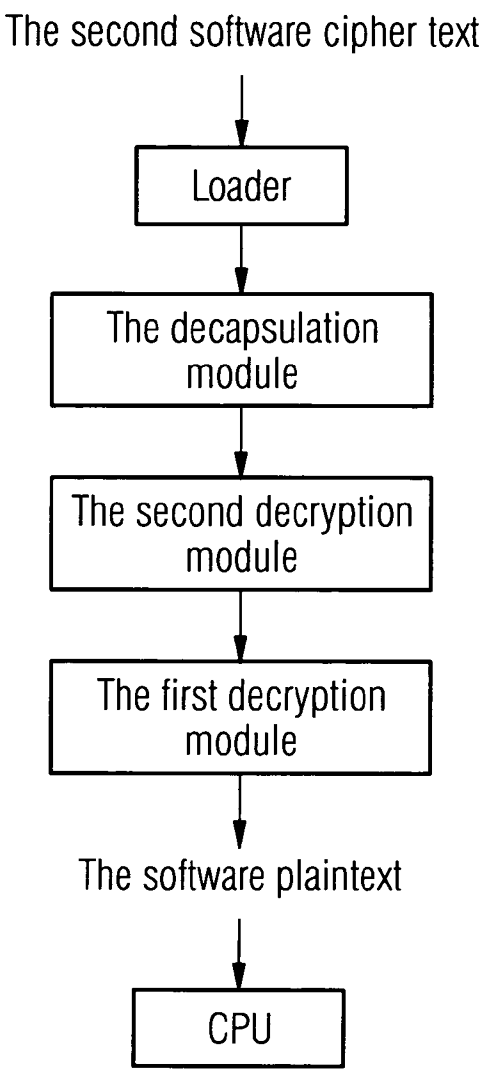

The second software cipher text
The decapsulation module
The second decryption module
The first decryption module
The software plaintext

FIG 5


The second software cipher text
Loader
The decapsulation module
The second decryption module
The first decryption module
The software plaintext
CPU

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2008/055912

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/22 H04L9/08

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 practical, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 915 025 A (TAGUCHI MASAHIRO [JP] ET AL) 22 June 1999 (1999-06-22) column 18, line 28 - column 20, line 33 figures 19-22 -----	1-9
A	MENEZES ET AL: "HANDBOOK OF APPLIED CRYPTOGRAPHY" HANDBOOK OF APPLIED CRYPTOGRAPHY, BOCA RATON, FL, CRC PRESS.; US, US, 1 January 1997 (1997-01-01), pages 567-570,546, XP002356115 ISBN: 978-0-8493-8523-0 page 550, paragraph 13.6 -----	1,5,8,9
A	WO 00/41357 A (NORTEL NETWORKS CORP [CA]; HARDJONO THOMAS P [US] NORTEL NETWORKS LTD) 13 July 2000 (2000-07-13) page 7, line 7 - page 8, line 25 ----- -/--	1-9

[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 document 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 22 July 2008	Date of mailing of the international search report 30/07/2008
Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Cartrysse, Kathy

International application No
PCT/EP2008/055912

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 02/25415 A (EDC SYSTEMS INC [US]; LEVINE RICHARD B [US]; LEE ANDREW R [US]; HOWARD) 28 March 2002 (2002-03-28) page 19, line 26 - page 20, line 3 claims 1-10 figure 8 -----	1-9
A	US 6 236 729 B1 (TAKARAGI KAZUO [JP] ET AL) 22 May 2001 (2001-05-22) column 23, line 64 - column 27, line 55 figure 16 -----	1-9

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2008/055912

Patent document cited in search report		Publication date	Patent family member(s)			Publication date
US 5915025	A	22-06-1999	JP	3627384	B2	09-03-2005
			JP	9258977	A	03-10-1997
WO 0041357	A	13-07-2000	AU	2960200	A	24-07-2000
			EP	1145480	A1	17-10-2001
			US	6182214	B1	30-01-2001
WO 0225415	A	28-03-2002	AU	9291001	A	02-04-2002
			AU	2001292910	B2	01-05-2008
			CA	2435624	A1	28-03-2002
			EP	1352307	A2	15-10-2003
US 6236729	B1	22-05-2001	NONE			