PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) **International Patent Classification** ^{6} :
G06F 13/00, 11/00, H04L 9/32, 9/00, H01L 9/08

A1

(11) **International Publication Number:** **WO 00/02132**

(43) **International Publication Date:** 13 January 2000 (13.01.00)

(21) **International Application Number:** PCT/US99/14434

(22) **International Filing Date:** 24 June 1999 (24.06.99)

(30) **Priority Data:**
09/109,472 2 July 1998 (02.07.98) US

(71) **Applicant** *(for all designated States except US)*: INTEL CORPORATION [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

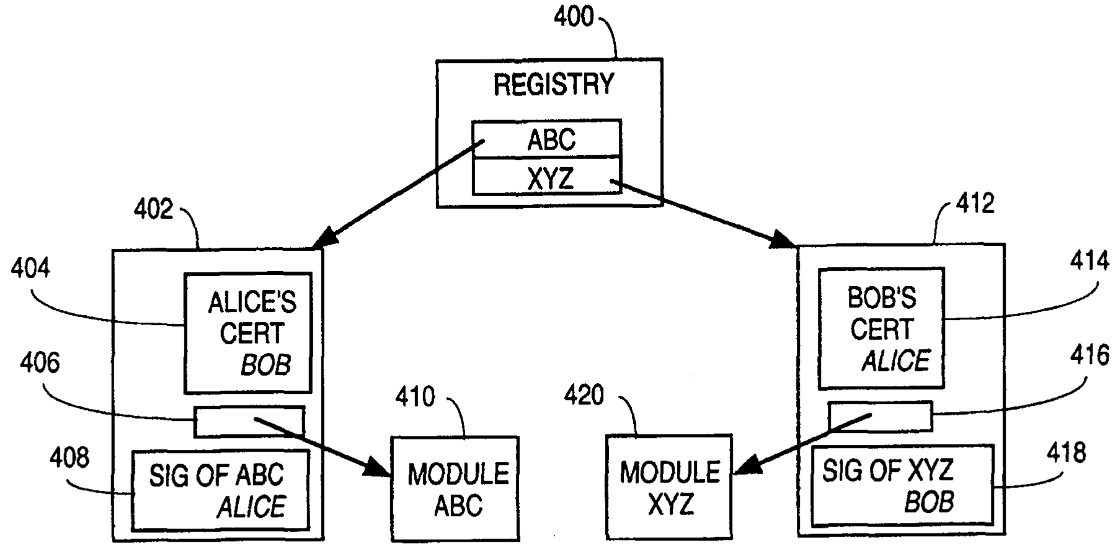
(72) **Inventors; and**
(75) **Inventors/Applicants** *(for US only)*: GRAUNKE, Gary, L. [US/US]; 12120 S.W. Trail Place, Beaverton, OR 97008-7010 (US). ROZAS, Carlos, V. [US/US]; 1534 N.W. Morgan Lane, Portland, OR 97229 (US).

(74) **Agents:** MILLIKEN, Darren, J. et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

(81) **Designated States:** AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) **Title:** METHOD AND APPARATUS FOR INTEGRITY VERIFICATION, AUTHENTICATION, AND SECURE LINKAGE OF SOFTWARE MODULES



(57) **Abstract**

A method and apparatus of authenticating and verifying the integrity of software modules (410, 420) is disclosed. In one embodiment, the software modules (400) initially establish their corresponding credentials. Then the local software module (410) ensures its integrity by validating its own digital signature (408). The local software module (410) having derived and validated certain information from the partner module's (404) credential. In addition, secure linkage between said local software module (410) and the partner software module (420) is maintained.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

METHOD AND APPARATUS FOR INTEGRITY VERIFICATION, AUTHENTICATION, AND SECURE LINKAGE OF SOFTWARE MODULES

FIELD OF THE INVENTION

This invention relates to data security generally and particularly to systems for providing data integrity verification and authentication functionality.

BACKGROUND OF THE INVENTION

As networked computer systems proliferate, individuals or corporations are subject to more and more unwanted attacks on their systems. For example, hackers have been reportedly successful in infiltrated into user's financial accounts and perform unauthorized transactions from the accounts. Further, even certain cryptographic applications have been proven vulnerable, because the trusted software is replaced with infected code without being detected. As a result, confidential information embedded in the application, such as the user's private key, is leaked out in the normal output of the application. Yet another type of attack is software viruses. These viruses infect their targets by masquerading as popular software or by attaching themselves to programs. When the target is infected, its confidential information may be compromised, or worse yet, its content may be destroyed.

In order to combat these attacks, users need methods of authenticating the origin of software and validating the integrity of the software. Many existing methods today address one aspect of the problem, but not the other. For instance, the known public-key cryptography and digital signature techniques are often compromised, because the private keys are often embedded in the software modules. Further, the authentication process often cannot proceed while the modules are being executed. Consequently, when these modules are in memory, they are susceptible to stealth virus attacks. As has been shown, an improved method and apparatus is needed to ensure not only the integrity of the software modules, but also the secured execution of such modules.

SUMMARY OF THE INVENTION

A method and apparatus of authenticating and verifying the integrity of software modules is disclosed. In one embodiment, each software module initially establishes its own credentials. Then the local software module ensures its integrity by validating its own digital signature. The local software module proceeds to authenticate the integrity of the partner software module after having derived and validated certain information from the partner module's credential. In addition, secure linkage between the local software module and the partner software module is maintained.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and is not limited by the figures of the accompanying drawings, in which like references indicate similar elements, and in which:

Figure 1 illustrates a public-key cryptographic system.

Figure 2 illustrates a digital signature system.

Figure 3 illustrates an example of two software modules applying public-key cryptography and digital signatures.

Figure 4 illustrates the block diagram of one embodiment of the present invention.

Figure 5 illustrates a general purpose computer system architecture.

Figure 6 illustrates a flow chart for a credential verifier used in partner module checking phase.

Figure 7 illustrates a flow chart for additional functionality of the credential verifier.

DETAILED DESCRIPTION

A method and apparatus for authenticating and verifying the integrity of software modules is described. In the following description, numerous specific details are set forth such as registry, etc. in order to provide a thorough understanding of the present invention. However, it will be apparent to one of ordinary skill in the art that the invention may be practiced without these particular details. In other instances, well known elements and theories such as public-key cryptography, encryption, digital signatures, cryptographic hash functions, etc. have not been discussed in special details in order to avoid unnecessarily obscuring the present invention. Also, unless having been indicated otherwise, words such as digital signature, digital certificate and software module are used interchangeably throughout the description with signature, certificate and module, respectively.

Figure 1 illustrates a known public-key cryptographic system. Msg 100 is un-encrypted binary data. It can be a stream of bits, a text file, a stream of digitized voice, or a digital video image, to present just a few of the many possibilities. Ciphertext 106 is the result of applying the encryption/decryption function, cipher 104, and the public key 102, to msg 100. In mathematical notation:

$\text{Cipher}_{104}(\text{Msg}_{100}, \text{Public Key}_{102}) = \text{Ciphertext}_{106}$

The receiver of ciphertext 106 can only recover msg 100 by applying cipher 104 and the proper private key 108 to ciphertext 106. In mathematical notation:

$\text{Cipher}_{104}(\text{Ciphertext}_{106}, \text{Private Key}_{108}) = \text{Msg}_{110}$
And the goal is to have:
$\text{Msg}_{100} = \text{Msg}_{110}$

A public-key cryptographic system is asymmetric. Specifically, the decryption key, or private key 108 is different than the encryption key, or public key 102. Although private key 108 cannot be readily derived from public key 102, they are related. Thus, without the proper public/private key pair, the receiver of ciphertext 106 will have a difficult time deciphering the encrypted msg 100. Additionally, the receiver is presumed to be the only person who knows his/her own private key 108, but public key 102 can be made public.

By applying similar encryption and decryption principles, digital signatures are often used to certify digital messages. Figure 2 illustrates such a known system. In particular, a unique value 204 for msg 200 is generated by means of one-way hash function 202. The resulting value 204 is considered a small finger print of msg 200. It also serves to deter malicious attackers from substituting false information for msg 200's original data. After val 204 is encrypted using private key 206 of the sender and signer of msg 200, the resulting ciphertext 210 is considered the digital signature of msg 200.

In order to verify that msg 200 has not been altered during transmission, the recipient first applies one-way hash function 202 to msg 218 and calculates val 214. Then the recipient compares val 214 with val 216. Val 216 is obtained by deciphering the signature, or ciphertext 210, using the signer's public key 212 with cipher 208.

Having discussed these fundamental encryption/decryption principles, Figure 3 demonstrates a prior art system with two software modules applying these principles. For example, module B only wishes to work with properly licensed modules, such as module A. Both modules A and B are presumed to know each others' public keys 306 and have their private keys 304 and 308 embedded in their own software. To start, module B issues a challenge 300 with a random number. Module A responds by signing the random number in response 302, or in mathematical notations:

R_1 = random number
$h_1 = \mathit{Hash}(R_1)$, where *Hash* is the previously discussed one-way hash function or 202 in Figure 2
E = encryption function or cipher 104 or 208 in Figures 1 and 2
Thus, response 302 = $E_{\text{private key 304}}(h_1)$, where private key 304 is used to encrypt h_1.

Since module B knows module A's public key, thus:
D = decryption function or cipher 104 or 208 in Figures 1 and 2
$h_2 = D_{\text{public key 306}}(E_{\text{private key 304}}(h_1))$

Then B performs another one-way hash function on the received random number, R_2, in response 302:
$h_3 = \mathit{Hash}(R_2)$
If h_2 equals to h_3, the identity of module A has been authenticated. Module B can then be assured that module A has been properly licensed. It should be noted that

the discussed authentication process may operate in a bilateral fashion. In other words, module A may initiate a challenge to authenticate module B.

However, this verification process for the software modules have several shortcomings. One, since each module's private key is embedded in the program, the key can be obtained by reverse engineering the module. Two, the integrity of the software modules is not verified in this process. Although known techniques can be used to verify module integrity when the module resides in secondary storage such as data storage device 510 in Figure 5, such techniques are incapable of validating the modules after they are loaded into primary memory such as random access memory 504 in Figure 5. In other words, when these modules reside in primary memory, they are susceptible to modifications. Three, although module B may believe that it has module A's public key and subsequently authenticates A's signature, module A may actually be module C and has defrauded module B into believing module C is module A.

Figure 4 illustrates the block diagram of one embodiment of the present invention which addresses all the mentioned flaws. The block diagram can be programmed or implemented in various types of systems. One such system is illustrated on a general purpose computer system in Figure 5. Such a computer system architecture comprises a bus element 500 for communicating information, a processor 502 coupled with said bus element 500 for processing audio, video and data information, a random access memory 504 coupled with said bus element 500 for storing static information and instructions for said processor 502, a data storage device 510 such as a magnetic disk and disk drive coupled with said bus element 500 for storing information and instructions, a readable medium 512 such as a floppy disk or a CD coupled with said bus element 500 for retrieving information and instructions for said processor 502, said bus element 500 coupled to network 514 for accessing resources on said network 514 such as server 518 and remote terminal 516, a display device 506 coupled to said bus element 500 for displaying information to the computer user and an alphanumeric or other type of input device 508 coupled to said bus element 500 for communicating information and command selections to said processor 502.

Operation of the Present Invention

The invention at hand is a method and apparatus for authenticating and verifying the integrity of software modules. Specifically, the present invention allows software modules to verify their data integrity even while the modules are being executed. Also, neither module contains the required secret information, such as private keys, for the authentication process. Lastly, since the present invention enables software modules to verify each other while they are being executed, the invention further ensures the modules' integrity by validating their expected memory locations and contents of the memory locations.

One embodiment of the invention is a general purpose computer system operating on various components as illustrated in Figure 4. Registry 400 points to software modules' credentials 402 and 412. Alice is the owner of software module ABC 410, and Bob is the owner of software module XYZ 420. When Alice initiates the authentication process to verify Bob's XYZ 420, Alice's ABC 410 is considered a *local* software module and Bob's XYZ 420 a *partner* software module. It should be noted that a local software module is not necessarily a calling module. For example, Alice's ABC 410 may call Bob's XYZ 420, but Bob's XYZ 420 may initiate the authentication process on Alice's module. In that scenario, even though Bob's XYZ 420 is a callee, it is still considered a local module in the following discussion. Moreover, when Alice's ABC 410 calls Bob's XYZ 420 or vice versa, a linkage between the modules is established.

Alice's credential 402 consists of Alice's certificate 404, a reference 406 to Alice's module ABC 410 and Alice's own signature 408 of module 410. Additionally, Alice's certificate 404 is signed by Bob using known digital signature schemes. On the other hand, Bob's credential 412 contains the same components as Alice's credential, namely, Bob's certificate 414, Bob's reference 416 to module XYZ 420, and Bob's signature 418 of module XYZ 420. Similarly, Bob's certificate 414 is signed by Alice.

It should be noted that although specific components in one embodiment of the present invention are described, other components may be incorporated without exceeding the scope of the invention. For example, Alice's credential 402 may include certificates of partners or licensing authorities, such as another digital certificate for Bob. This added certificate may be issued by Bob's company in an attempt to attest to Bob's authority to sign Alice's certificate 404. Furthermore, Alice's module ABC 410 may want to cooperate with modules other than XYZ 420. Thus, registry 400 may include additional entries, and Alice's credential 402 may encompass additional certificates signed by Alice's desired partners.

The operations of the illustrative system in Figure 4 can be characterized in four phases: credential building phase, credential registering phase, self checking phase and partner module checking phase. In one embodiment of the present invention, each owner of a software module has a credential builder. Using the example in Figure 4, in phase 1 or the credential building phase, the credential builder takes various inputs such as Alice's certificate signed by Bob, a reference to module ABC and Alice's signature of module ABC and forms Alice's credential. An example of a resulting credential for Alice is a computer file with all the recited input information.

In phase 2 or the credential registering phase, one embodiment is to register Alice's credential in a predetermined location, such as registry 400, so that other software modules know where to look for Alice's module information. It should be noted that registry 400, Alice's and Bob's credentials and module information in Figure 4 do *not* all have to reside on the same machine. For instance, Alice's credential 402 may reside on a network server. When Bob's module attempts to find Alice's module, registry 400 will direct Bob to the appropriate server on the network to identify Alice's credential 402. After having located Alice's credential 402, reference 406 within credential 402 then points Bob to Alice's module ABC 410.

Also, although the registry is used in describing one embodiment of the present invention, it will be apparent to one of ordinary skill in the art to use other methods for tracking and locating software modules' credentials. Finally, as the embodiment in Figure 4 demonstrates, software modules and their credentials are separate entities. Therefore, software modules do not have any secret information embedded in them, such as their own private keys.

After credentials have been established and registered, software modules then enter phase 3 or the self checking phase. Again using Figure 4 as an illustration, when Alice's module is initializing, Alice first searches through registry 400 for her own credential. After the registry points her to credential 402, Alice uses her digital signature verifier with her public key to verify her signature 408. Specifically, the digital signature verifier is described in detail above and in connection with Figure 2. In mathematical notations:

$h_1 = \textit{Hash}(\text{Alice's module}_{t1})$, where *Hash* is the previously discussed one-way hash function or 202 in Figure 2, and $\text{Alice's module}_{t1}$ stands for a snap shot of the module at time one.

E = encryption function or cipher 104 or 208 in Figures 1 and 2

Thus, Alice's signature 408 = $E_{\text{Alice's private key}}(h_1)$, where Alice's private key is used to encrypt h_1.

Since Alice knows her own public key, thus:

D = digital signature verifier or cipher 104 or 208 in Figures 1 and 2

$h_2 = D_{\text{Alice's public key}}(E_{\text{Alice's private key}}(h_1))$

Then Alice may perform another one-way hash function on her current module:

$h_3 = \mathit{Hash}(\text{Alice's module}_{t2})$ where t2 stands for time two and occurs later than time one.

If h_2 equals to h_3, then the integrity of Alice's module has been authenticated. Or in other instances, h_3 may not need to be computed. But rather, Alice simply needs to compare h_1 with h_2.

Although the illustrated digital signature verifier uses one type of message recovery signature scheme in phase 3, it should be apparent to one ordinarily skilled in the art to apply other digital signature schemes without exceeding the scope of the present invention. One such scheme directly uses signature 210, public key 212 and hash value 214 in one verification step. Furthermore, even though the above mathematical equations calculate Alice's signature 408 using Alice's entire module at time 1, or $E_{\text{Alice's private key}}(h_1)$, it should be apparent to one ordinarily skilled in the art to generate signature 408 with only portions of Alice's module. For instance, the signature generation process may omit the initialized data sections of Alice's module, since the data residing in these sections change during module's execution.

It should also be emphasized that Alice's module contains only her public key information and not her private key information. It should be further noted that this self checking phase can be performed on a memory image of Alice's module in addition to other machine readable mediums such as CDs or disks. With the capability of verifying memory images of software modules, when modules reside in primary memory such as random access memory 504 in Figure 5, the present invention may continuously validate the modules even when the modules are being executed.

After Alice's module has been verified, Alice's module calls Bob's module and initiates the authentication process. Therefore, Alice's module is both the caller module and the local module. Bob's module, on the other hand, is both the callee module and the partner module. Figure 6 details a credential verifier's flow chart in phase 4 or the partner module checking phase in one embodiment of the present invention. It should be noted that some steps of the credential verifier can either be expanded, eliminated or combined with other steps without exceeding the scope of the invention.

Using Figure 4 as an illustration, in step 600 shown in Figure 6, Alice's module or the local module searches through registry 400 to search for the reference to Bob's credential. After finding Bob's credential 412, Alice proceeds in step 602 to identify Bob's certificate 414, signed by Alice, in Bob's credential 412. Since Alice knows her own public key, in step 604, Alice puts her public key along with her signature on Bob's certificate 414 into a digital signature verifier, which has been discussed extensively above and in connection with Figure 2.

When the verification result from step 606 is invalid, step 608 signals an error to Alice and stops Alice from proceeding further. If however the result of the comparison in step 606 is valid, Alice can now trust Bob's certificate 414. Since Bob's certificate 414 contains Bob's public key information, Alice obtains Bob's public key in step 610. With Bob's public key, Alice proceeds to find Bob's signature 418 of Bob's software module 420 in step 612. In step 614, Alice enters her copy of Bob's public key and Bob's signature 418 into a digital signature verifier. If the result in step 616 is invalid, step 620 again signals an error to Alice and stops Alice from continuing on. On the other hand, a valid result indicates that Bob's module has been authenticated, and Alice may proceed to work with Bob's module. It should be emphasized that the present invention can operate in a bilateral fashion. Thus, Bob can similarly authenticate Alice's module utilizing all the described steps.

It should be further noted that the described credential verifier can continue operating in part or as a whole while Alice's and Bob's modules are being executed. This ensures the integrity of the modules during the modules' execution. In particular, depending on design parameters and goals, certain steps shown in Figure 6 may not need to execute each time the credential verifier is invoked. Each software module may also determine its required frequency of invoking the credential verifier. These mentioned variations in implementing and invoking the credential verifier should be apparent to one of ordinary skill in the art without departing from the spirit and the scope of the present invention.

Moreover, other verification schemes can be included in the credential verifier without exceeding the scope of the present invention. For example, step 614 may incorporate additional checking policies to raise the level of authenticity for the modules. In particular, Figure 7 illustrates one such policy. In step 700, Alice checks whether Bob's module has been loaded into memory. If not, in step 702 Alice verifies Bob's modules residing on other machine readable mediums such as disks, CDs, etc. using the previously described digital signature verifier. If the result from step 703 is valid, Alice proceeds to either wait for Bob to load his own module into memory or loads the module into memory for Bob in step 704. An invalid result from step 703 directs Alice to step 710 and stops her from continuing on.

However, when Bob's module has been loaded into memory, Bob's module is verified using the digital signature verifier in steps 706 and 708. After the validation of Bob's module, Alice performs another check in step 712 to ensure a secured linkage between Alice's module and Bob's module. In particular, a memory checker checks the addresses to be called, or the destination addresses in Bob's module, against the valid address ranges of Bob's module. If the destination addresses remain within the valid ranges in step 712, Alice can proceed to work with Bob's module. Additionally, the memory checker checks a caller's addresses against the calling module's, or Alice's module, valid address ranges. As has been mentioned previously, such verification steps can continue while the modules are being executed.

Since an operating system's program loader may modify software modules when the loader moves the modules from secondary memory such as data storage device 510 shown in Figure 5 to primary memory such random access memory 504 shown in Figure 5, the memory checker may include intelligence to reverse such modifications. In particular, a program loader often modifies program instructions to reflect the address changes from the secondary memory to the primary memory. The memory checker in one embodiment of the invention may elect to copy the affected regions of the modules from the primary memory, reverse the address modifications and then attempt to reconstruct and validate the signed image of the module.

Thus, a method and apparatus for authenticating the integrity of software modules has been disclosed. Although the present invention has been described particularly with reference to the figures, it will be apparent to one of the ordinary skill in the art that the present invention may appear in any number of systems which provide data integrity verification and authentication functionality. It is further contemplated that many changes and modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the present invention.

CLAIMS

What is claimed is:

1. A method of authenticating and verifying the integrity of software modules, the method comprising the steps of:
    a. establishing a credential for a local software module and a credential for a partner software module;
    b. verifying the integrity of said local software module by validating said local software module's digital signature;
    c. authenticating the integrity of said partner software module by deriving and validating information from said partner software module's credential; and
    d. securing linkage between said local software module and said partner software module.

2. The method according to claim 1, wherein said local and said partner software modules reside on a machine readable medium.

3. The method according to claim 1, wherein said establishing step further includes the steps of:

providing a digital signature of said local software module in said local software module's credential;

providing a digital certificate of said local software module issued by said partner software module in said local software module's credential;

providing a reference to said local software module in said local software module's credential;

providing a digital signature of said partner software module in said partner software module's credential;

providing a digital certificate of said partner software module issued by said local software module in said partner software module's credential; and

providing a reference to said partner software module in said partner software module's credential.

4. The method according to claim 1, wherein step (c) further comprises:

identifying said credentials;

identifying and validating said certificates within said credentials; and

identifying and validating said digital signatures within said credentials.

5. The method according to claim 2, wherein said software modules' corresponding credentials reside external to said software modules.

6. The method according to claim 4, wherein step (d) further comprises:

having a machine execute said local and partner software modules in said machine's memory locations; and

validating said memory locations and contents of said memory locations.

7. An apparatus for authenticating and verifying the integrity of software modules, the apparatus comprising:

a. means for establishing a credential for a local software module and a credential for a partner software module;
b. means for verifying the integrity of said local software module by validating said local software module's digital signature;
c. means for authenticating the integrity of said partner software module by deriving and validating information from said partner software module's credential; and
d. means for securing linkage between said local software module and said partner software module.

8. The apparatus according to claim 7, wherein said local and said partner software modules reside on a machine readable medium.

9. The apparatus according to claim 7, wherein said establishing means further comprises:

providing a digital signature of said local software module in said local software module's credential;

providing a digital certificate of said local software module issued by said partner software module in said local software module's credential;

providing a reference to said local software module in said local software module's credential;

providing a digital signature of said partner software module in said partner software module's credential;

providing a digital certificate of said partner software module issued by said local software module in said partner software module's credential; and

providing a reference to said partner software module in said partner software module's credential.

10. The apparatus according to claim 7, wherein (c) further comprises:

means for identifying said credentials;

means for identifying and validating said certificates within said credentials; and

means for identifying and validating said digital signatures within said credentials.

11. The apparatus according to claim 8, wherein said software modules' corresponding credentials reside external to said software modules.

12. The apparatus according to claim 10, wherein said securing means further performs the steps of:

executing said local and partner software modules in said apparatus's memory locations; and

validating said memory locations and contents of said memory locations.

13. An apparatus for authenticating and verifying the integrity of software modules, the apparatus comprising:

a. a credential builder for establishing a credential for a local software module and a credential for a partner software module;

b. a digital signature verifier for verifying the integrity of said local software module by validating said local software module's digital signature;

c. a credential verifier for authenticating the integrity of said partner software module by deriving and validating information from said partner software module's credential; and

d. a memory checker for securing linkage between said local software module and said partner software module.

14. The apparatus according to claim 13, wherein said local software and said partner software modules reside on a machine readable medium.

15. The apparatus according to claim 13, wherein said credential builder further comprises:

providing a digital signature of said local software module in said local software module's credential;

providing a digital certificate of said local software module issued by said partner software module in said local software module's credential;

providing a reference to said local software module in said local software module's credential;

providing a digital signature of said partner software module in said partner software module's credential;

providing a digital certificate of said partner software module issued by said local software module in said partner software module's credential; and

providing a reference to said partner software module in said partner software module's credential.

16. The apparatus according to claim 13, wherein said credential verifier further comprises:

registry entries for referencing to said credentials;

a certificate verifier for validating said certificates within said credentials; and

a signature verifier for validating said digital signatures within said credentials.

17. The apparatus according to claim 14, wherein said software modules' corresponding credentials reside external to said software modules.

18. The apparatus according to claim 16, wherein said memory checker further performs the steps of:

executing said local and partner software modules in said apparatus's memory locations; and

validating said memory locations and contents of said memory locations.

19. A machine readable medium having embodied thereon instructions, which when executed by a machine, causes said machine to:

authenticate the integrity of software modules on said medium by

a. establishing a credential for a local software module and a credential for a partner software module;

b. verifying the integrity of said local software module by validating said local software module's digital signature;

c. authenticating the integrity of said partner software module by deriving and validating information from said partner software module's credential; and

d. securing linkage between said local software module and said partner software module.

20. The machine readable medium according to claim 19, wherein said establishing step further includes the steps of:

providing a digital signature of said local software module in said local software module's credential;

providing a digital certificate of said local software module issued by said partner software module in said local software module's credential;

providing a reference to said local software module in said local software module's credential;

providing a digital signature of said partner software module in said partner software module's credential;

providing a digital certificate of said partner software module issued by said local software module in said partner software module's credential; and

providing a reference to said partner software module in said partner software module's credential.

21. The machine readable medium according to claim 19, wherein step (c) further comprises:

identifying said credentials;

identifying and validating said certificates within said credentials; and identifying and validating said digital signatures within said credentials.

22. The machine readable medium according to claim 19, wherein said software modules' corresponding credentials reside external to said software modules.

23. The machine readable medium according to claim 21, wherein step (d) further comprises:

executing said local and partner software modules in said machine's memory locations; and

validating said memory locations and contents of said memory locations.

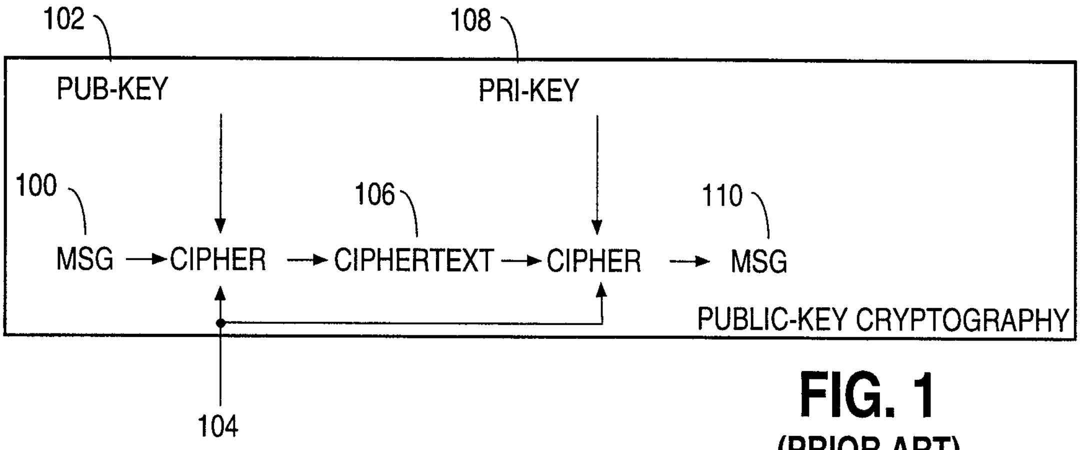


FIG. 1
(PRIOR ART)

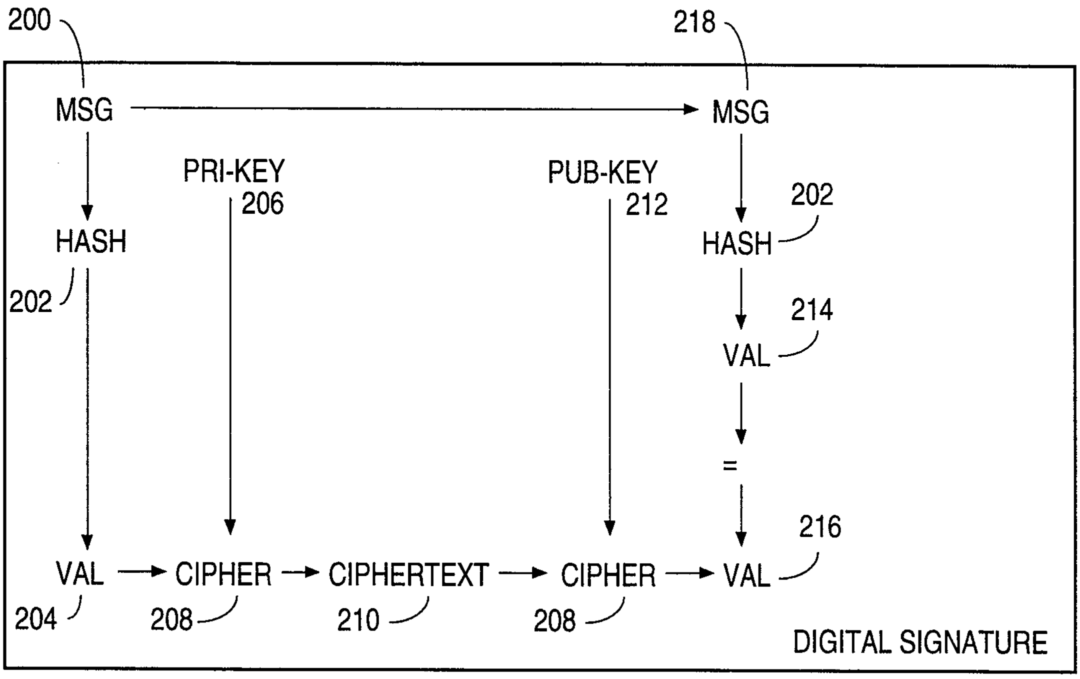


FIG. 2
(PRIOR ART)

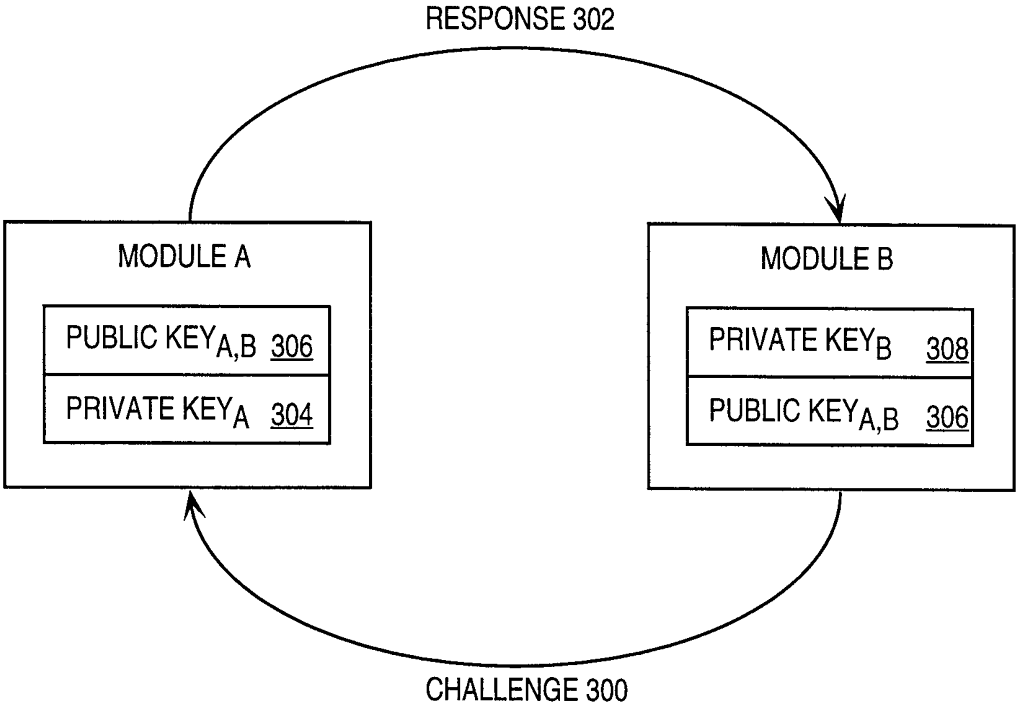


FIG. 3
(PRIOR ART)

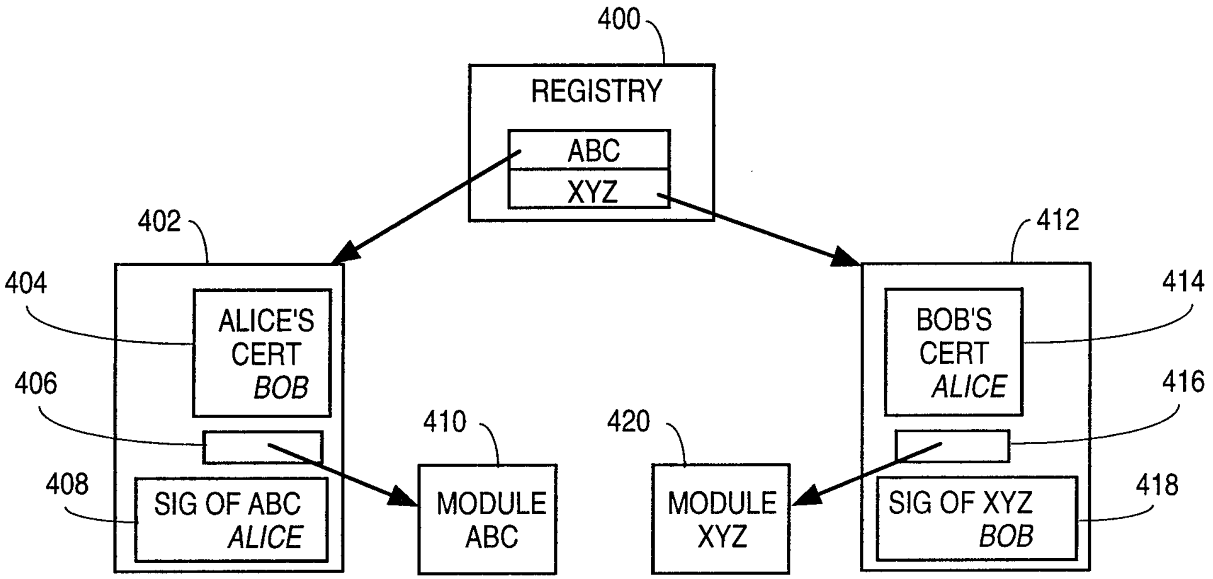
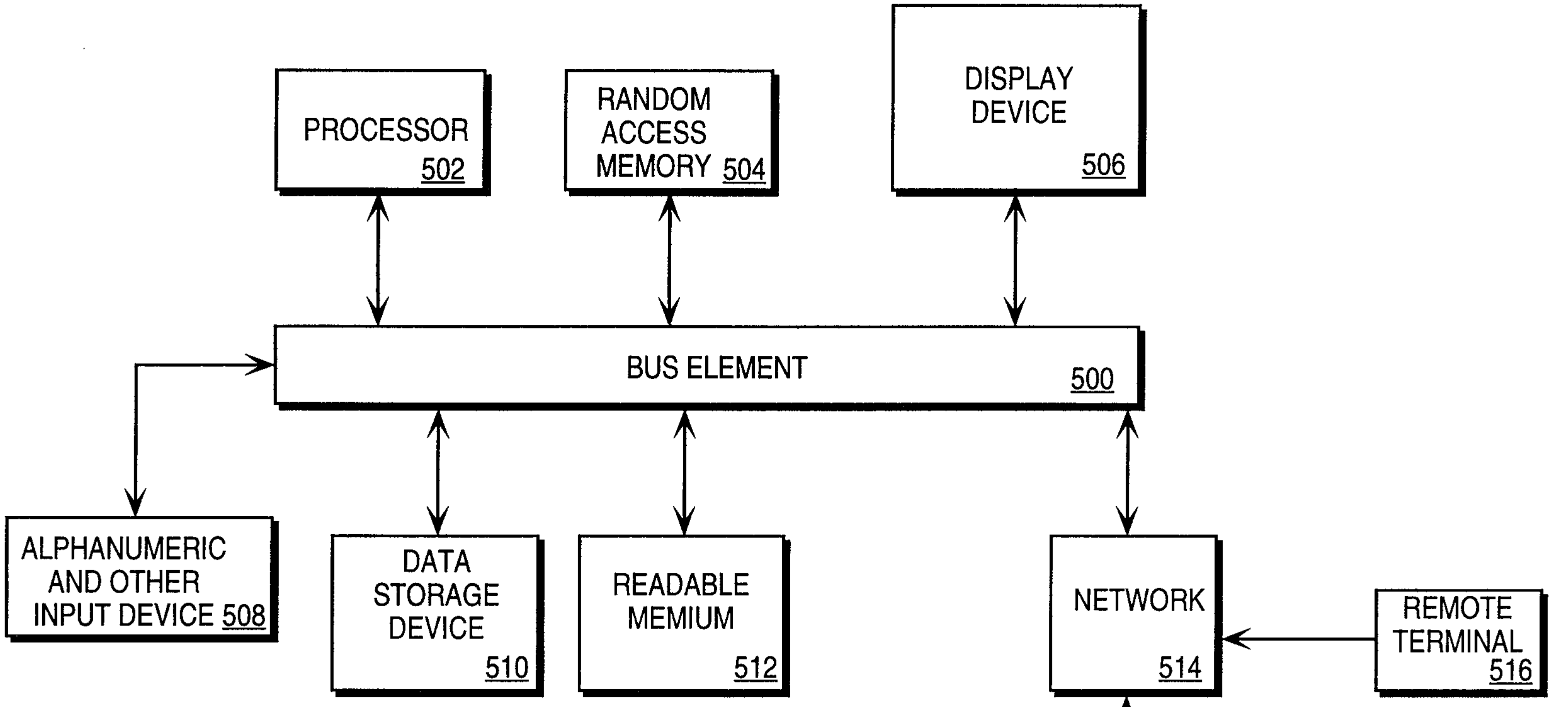

400
REGISTRY
ABC
XYZ
402
404
ALICE'S
CERT
BOB
406
408
SIG OF ABC
ALICE
410
MODULE
ABC
420
MODULE
XYZ
412
414
BOB'S
CERT
ALICE
416
418
SIG OF XYZ
BOB

FIG. 4


PROCESSOR
502
RANDOM
ACCESS
MEMORY 504
DISPLAY
DEVICE
506
BUS ELEMENT
500
ALPHANUMERIC
AND OTHER
INPUT DEVICE 508
DATA
STORAGE
DEVICE
510
READABLE
MEMIUM
512
NETWORK
514
REMOTE
TERMINAL
516

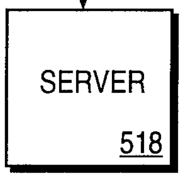

SERVER
518

FIG. 5

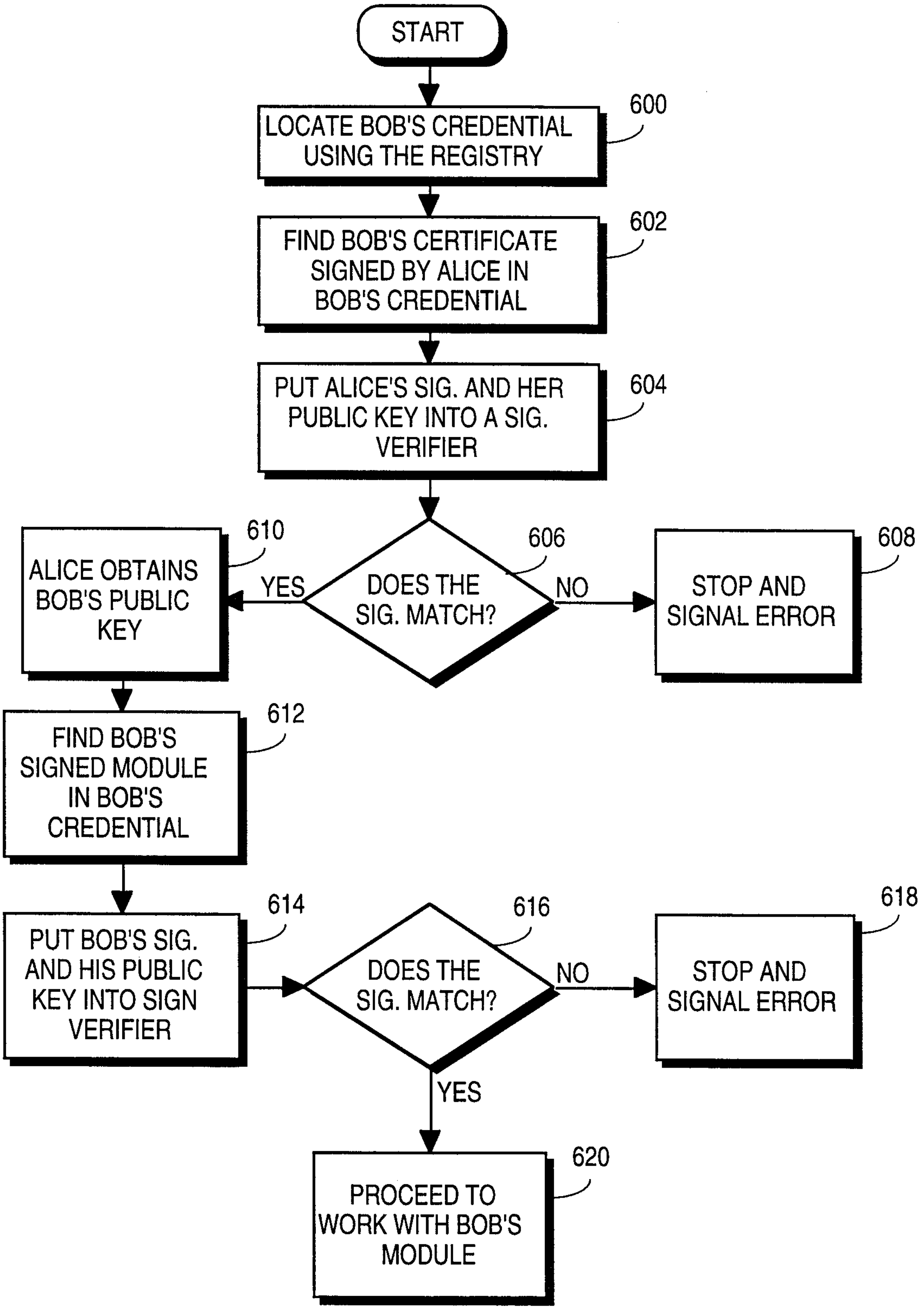

START
LOCATE BOB'S CREDENTIAL USING THE REGISTRY
600
FIND BOB'S CERTIFICATE SIGNED BY ALICE IN BOB'S CREDENTIAL
602
PUT ALICE'S SIG. AND HER PUBLIC KEY INTO A SIG. VERIFIER
604
DOES THE SIG. MATCH?
606
YES
NO
ALICE OBTAINS BOB'S PUBLIC KEY
610
STOP AND SIGNAL ERROR
608
FIND BOB'S SIGNED MODULE IN BOB'S CREDENTIAL
612
PUT BOB'S SIG. AND HIS PUBLIC KEY INTO SIGN VERIFIER
614
DOES THE SIG. MATCH?
616
NO
STOP AND SIGNAL ERROR
618
YES
PROCEED TO WORK WITH BOB'S MODULE
620

FIG. 6

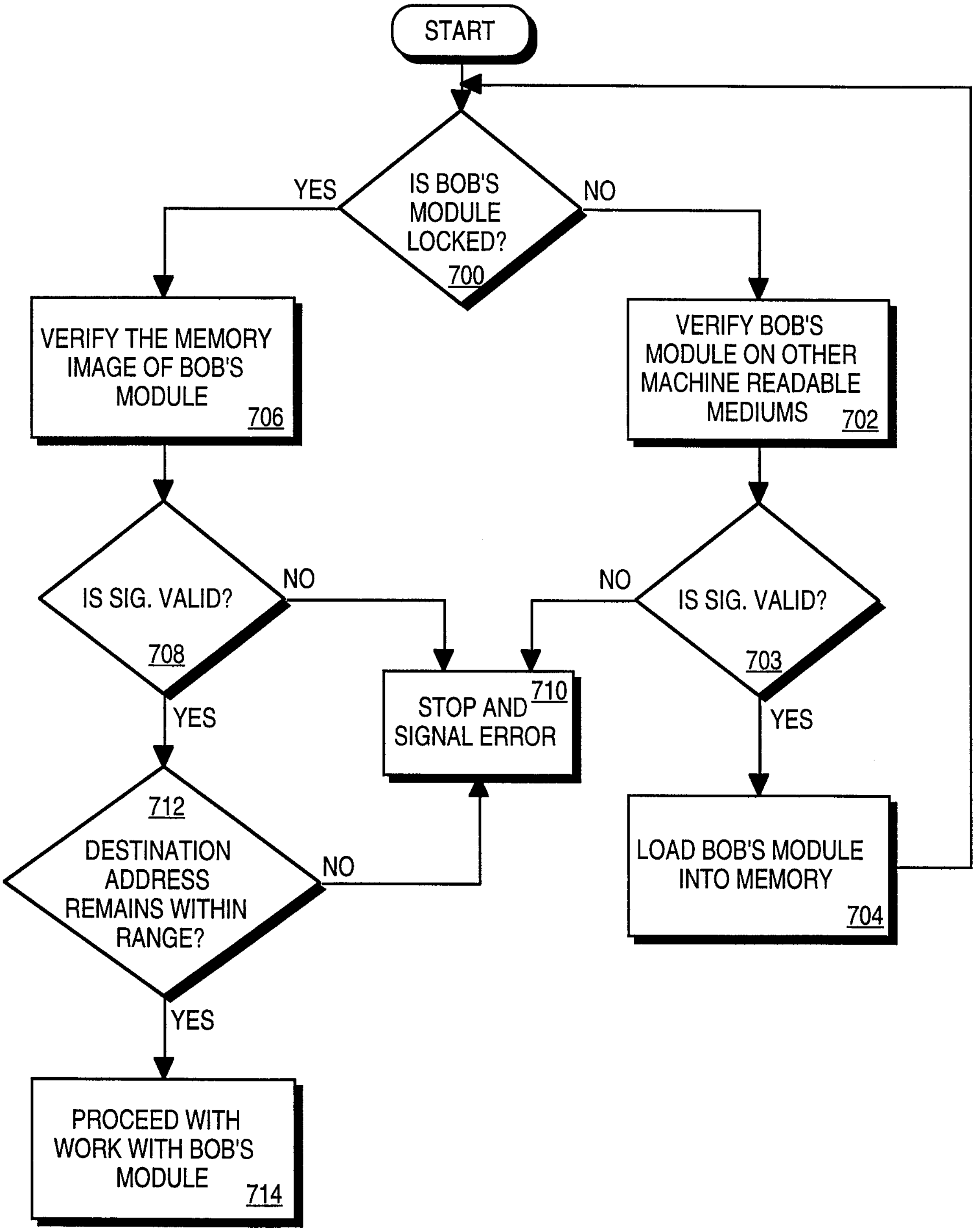

START
IS BOB'S MODULE LOCKED?
700
YES
NO
VERIFY THE MEMORY IMAGE OF BOB'S MODULE
706
VERIFY BOB'S MODULE ON OTHER MACHINE READABLE MEDIUMS
702
IS SIG. VALID?
708
NO
NO
IS SIG. VALID?
703
YES
STOP AND SIGNAL ERROR
710
YES
712
DESTINATION ADDRESS REMAINS WITHIN RANGE?
NO
LOAD BOB'S MODULE INTO MEMORY
704
YES
PROCEED WITH WORK WITH BOB'S MODULE
714

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/14434

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) : GO6F 13/00, 11/00; HO4L 9/32, HO4L 9/00; HO1L 9/08
US CL : 713/200, 201; 380/3-4, 21, 23, 25; 705/39
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)

U.S. : 713/200, 201; 380/3-4, 21, 23, 25; 705/39

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

N/A

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS, JPO, STN- INPADOC, COMPENDEX, INSPEC search terms: program, modules, switching, exchange, swapping, software, verified, integrity, authenticate, partner, certificate.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,559,814 A (ROLIN ET AL.) 24 September 1996, see entire document.	1-2,7-8,13-14,19-22
Y	US 5,625,693 A (ROHATGI ET AL.) 29 April 1997, see entire document.	1-2,7-8,13-14,19-21
X	US 5,757,914 A (McMANIS) 26 May 1998, see entire document.	1-2,7-8,13- 14
Y,E	US 5,933,503 A (SCHELL ET AL.) 03 August 1999, see entire document.	1-23
Y,P	US 5,901,227 A (PERLMAN) 04 May 1999, see entire document.	1-23
A,P	US 5,903, 721 A (SIXTUS) 11 May 1999, see entire document.	1-23

☒ 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 document 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 20 SEPTEMBER 1999	Date of mailing of the international search report 22 OCT 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer NORMAN MICHAEL WRIGHT James R. Matthews Telephone No. (703) 308-0000

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,P	US 5,915,021 A (HERLIN ET AL.) 22 June 1999, see entire document.	1-23
Y,P	US 5,872,848 A (ROMNEY ET AL.) 14 Feburary 1999, see entire document.	1-23