EXECUTABLE SOFTWARE SECURITY SYSTEM

Inventor: Mark Ellery Ogram, Tucson, AZ (US)

Correspondence Address:
Mark E. Ogram
7454 E. Broadway # 203
Tucson, AZ 85710 (US)

Related U.S. Application Data

Continuation-in-part of application No. 11/170,229, filed on Jun. 28, 2005.

Publication Classification

INT. Cl. G06F 12/14 (2006.01) U.S. Cl. 713/189

ABSTRACT

A computer system which is configured to load executable programs. This configuration first accepts an operator defined key; withdraws an encrypted executable program from memory; and, using the operator defined key, decrypts the encrypted executable program into a functional executable program. It is this functional executable program which is used by the processing unit. During shutdown, each executable program is checked to see if it was derived from an encrypted executable program; those that aren’t, are verified as being legitimate by the operator prior to their storage into the memory.
Fig 5

E-mail Response

Verification

Fig 6

Secure Program Operation
Fig 7
remainder
sub routine
R(A,B)

70A \rightarrow \text{BEGIN}

71A \rightarrow P = 0

72 \rightarrow A, B

73A \rightarrow A < B ?

73B / \rightarrow \text{loop}

71C \rightarrow \text{if}

71D \rightarrow C = B - (P-1) \times A

RETURN C
Encryption

\[ E_J = R\left[ o_J + R(s_J, m_J), m_J \right] \]
Fig 9

START \( s_{90A} \)

\( E \neq F \) \( ? \) \( s_{91A} \)

\( E_j \) \( s_{92A} \)

\( S_j \times X_j \) \( s_{92B} \)

\( C = \frac{R(S_j \times X_j)}{s_{93}} \)

\( C \leq E_j \) \( ? \) \( s_{94A} \)

\( O_j = E_j - C \)

\( N \) \( s_{94B} \)

\( O_j = E_j + C - X_j \)

\( O_j \) \( s_{95} \)
Fig 11A
MAIL OPERATION
SENDING

Fig 11B
MAIL OPERATION
RECEIVING

START

DESIGNATION
MESSAGE

KEY #

ENCRYPT MESSAGE

DESTINATION
WITH ENCRYPTED MESSAGE

STOP

START

SOURCE OF
ENCRYPTED MESSAGE

KEY #

DECRYPT ENCRYPTED MSG

MSG

STOP
FIGURE 12

<table>
<thead>
<tr>
<th>J</th>
<th>Oj</th>
<th>Si</th>
<th>Value</th>
<th>Value</th>
<th>Ei</th>
<th>Letter</th>
<th>Value</th>
<th>Decrypted Letter</th>
<th>Oj</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>t</td>
<td>20</td>
<td>4</td>
<td>24</td>
<td>x</td>
<td>20</td>
<td>t</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>h</td>
<td>8</td>
<td>20</td>
<td>2</td>
<td>b</td>
<td>8</td>
<td>h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>e</td>
<td>5</td>
<td>6</td>
<td>11</td>
<td>k</td>
<td>5</td>
<td>e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>21</td>
<td>21</td>
<td>21</td>
<td>u</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>r</td>
<td>18</td>
<td>22</td>
<td>14</td>
<td>n</td>
<td>18</td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>c</td>
<td>5</td>
<td>39</td>
<td>18</td>
<td>r</td>
<td>5</td>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>d</td>
<td>4</td>
<td>27</td>
<td>5</td>
<td>e</td>
<td>4</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>43</td>
<td>22</td>
<td>4</td>
<td>h</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>d</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>v</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>o</td>
<td>15</td>
<td>14</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>o</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>g</td>
<td>7</td>
<td>32</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>g</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>r</td>
<td>18</td>
<td>81</td>
<td>21</td>
<td>18</td>
<td>18</td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>a</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>n</td>
<td>14</td>
<td>17</td>
<td>5</td>
<td>14</td>
<td>14</td>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>h</td>
<td>8</td>
<td>14</td>
<td>22</td>
<td>8</td>
<td>h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>o</td>
<td>15</td>
<td>42</td>
<td>5</td>
<td>15</td>
<td>o</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>m</td>
<td>13</td>
<td>8</td>
<td>21</td>
<td>13</td>
<td>m</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>c</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fig 14

START 5/40A

KEY

DECRYPT
ENCRIPTED
TEMPLATE

INFORM
OPERATOR

1/41

KEY

VERIFIED?

Y

STOP 5/40B

N

1/43

1/42
Fig 15A

DECRYPTED TEMPLATE

AUTHORIZATION

AUTHORIZED

? 152A

Fig 15B

TEMPLATE IDENTICAL TO KEY?

Y
Comparison Template

Comparison identical to decrypted template?

Y

Fig 15c
160A — START

161A — WITHDRAW ENCRYPTED PROGRAM

161B — DECRYPT PROGRAM

161C — PROGRAM TO PROCESSING UNIT

160B — STOP

Fig 16
START

IDENTIFY
EXECUTABLE PROGRAM

COMPARABLE
ENCRYPTED PROGRAM?

REPLACE
EXECUTABLE?

MORE?

STOP

STORAGE?

ENCRYPT PROGRAM

STORE ENCRYPTED PROGRAM

Fig 17
EXECUTABLE SOFTWARE SECURITY SYSTEM

[0001] This is a continuation-in-part of U.S. patent application Ser. No. 11/170,229, filed on Jun. 28, 2005, and entitled, “Encrypted Communications”.

BACKGROUND OF THE INVENTION

[0002] This invention relates generally to the communication of data and more particularly to communications which are encrypted.

[0003] While distributed network systems such as the Internet, have expanded the horizons for the world in the collection and dissemination of knowledge, by the very nature of these systems, there has developed a growing awareness that information which is so easily obtained, is also lost with the same ease. The problems and crimes associated with the broad dissemination of information have become commonplace occurrences, and the problems are only expected to become more pronounced in the future.

[0004] These problems include such things as: identity theft; credit card theft; hacking into private data-bases; disrupting private computers through “viruses”; disruption of governmental data bases; fraudulent control of traffic systems; and many more.

[0005] Central to all of these problems is the intrinsic anonymous nature of the communications. A receiver of information receives only bits/bytes of digital information and the source of such digital information is generally unknown. Within the Internet, identities are easily created.

[0006] In an attempt to provide some level of knowledge of the other side, passwords and ID’s (identification values/symbols) are often used. Unfortunately, often these passwords/IDs are stolen and are then used indiscriminately by a criminal or hacker.

[0007] Another technique which has been used to curtail the improper gathers of information is the creation of encryption techniques such as the iKP protocol. These protection schemes though attempt to develop a standard encryption methodology which is used for every secure transmission, but this requirement in and of itself tends to make the encryption both difficult in use and in storing.

[0008] Almost by the very nature of encryption, encryption must be complex. The Enigma Machine developed by Germany during World War II was an elaborate and complex systems of gears which was used to map each new character and which relied upon the previously mapped message in determining how the next character was mapped.

[0009] While there is a natural tendency to use “complex” solutions, these complexities make the use of the solution difficult if not impossible.

[0010] Another problem which computer users have encountered is the unauthorized planting of viruses, “spyware”, and other programs into a user’s computer. These unauthorized programs often enter the computer innocuously during normal operation of the computer and are then stored into the computer’s memory automatically during normal shut-down of the computer.

[0011] If left unchecked, these unauthorized programs can cripple a computer, and in some situations, sensitive data is stolen without the user ever being aware of the theft.

[0012] It is clear there is a need for an efficient protection from the unauthorized use of an individual’s computer.

SUMMARY OF THE INVENTION

[0013] A communications system in which a sending computer encrypts a message using a key associated with the computer which is to receive the message; the receiving computer uses a key associated with the sending computer in the decryption process.

[0014] In the preferred embodiment, the sending computer is equipped with a set of keys and each key within the set is usable for the encryption process. The selection of a particular key depends on the destination of the message; or, if it is the first time a message is being sent to that destination, the key is arbitrarily selected and a record associated the arbitrarily selected key and the destination is made for future reference.

[0015] While the present discussion refers to “computer”, the invention is not intended to apply solely to a single or stand-alone computer. Rather, the term “computer” is intended to relate to a single computer as well as a system of computers which work in concert to obtain the objectives outlined.

[0016] The following discussion recognizes that a computer is configured to perform a designated operation on data to obtain a desired result. Configuration of a computer is often done through a programming language (e.g. assembly, basic, Colbol, Fortran, C,) which defines the function of the computer; but, in some situations, “hard wired” or dedicated circuitry is also used.

[0017] Within the present discussion, the invention relates to a sequence of symbols which are represented in a digital manner. Those of ordinary skill in the art readily recognize a variety of such sequences such as the American Standard Code for Information Interchange (ASCII). In some situations, the digital map to symbols is arbitrarily done. In this case, each symbol is arbitrarily assigned a unique value which forms another level of encryption.

[0018] The present discussion refers to the Internet, but, the invention is not intended to be so limited and is viable for any distributed network of computers.

[0019] For ease in reference, some many of the terms used herein, such as “computers”, “keys”, “data”, “messages” and the like, have been given labels (such as first, second third or primary, secondary, etc.) to help identify them; but, these labels are not intended to be limiting as to the order of use, ownership, or physical position.

[0020] Within this invention, each “computer” is defined by its capabilities or function.

[0021] Within the present invention, each digital value which is to be communicated, is mapped uniquely to another value within the field. In this manner, the mapping or encrypting is done on an individual value without any necessary reference to prior or future encryptions. To accomplish this unique mapping objective, the encrypting site and the decrypting site both have a “key” which is used both for the encrypting and decrypting operation. Since the “key” or mapping template provides a unique mapping and that “key” is not available to others, the possibility of a “hacker” being able to fraudulently decrypt the message is all but eliminated.
0022. In this context, the “key” is a series of values which are used in both the mapping process and the reverse-mapping process and consists of a series S_j.

0023. The creation of the key is accomplished through a variety of techniques, including, but not limited to: random number generation, prior data based, fixed set, historically based, based on the computer identification/serial number, or any combination of the above.

0024. Random number based keys are created using a programmed or “canned” random number generator. These generators produce a series of values which appear random, but in actuality are not truly random in that each time the random number generation program is initiated, it produces an identical series of “random numbers”; hence, if the encrypting and the decrypting computers operate the same random number generator, both computers develop identical series of values.

0025. An alternative technique creates a series of numbers to create the key using values from the message or the key itself which have been produced or provided earlier. In this case, a Markov type of series is produced. The creation of the function which produces this series of values is limitless and relies only upon the creative power of the developer. As example, the following are all possible functions:

\[ S_{j+1} = 3S_{j} \oplus 2S_{j-1} \quad \text{OR} \quad S_{j+1} = 2S_{j} \oplus 3S_{j-1} \]
\[ S_{j+1} = 3O_{j} \oplus 2O_{j-1} \]
\[ S_{j+1} = 5O_{j} \oplus 3O_{j-1} \]
\[ S_{j+1} = 5O_{j} \oplus 5O_{j-1} \]
\[ S_{j+1} = 5 + 2S_{j} \]
\[ S_{j+1} = 5 + 2O_{j} \]

(Note, within this discussion, “*” denotes multiplication; “ABS” denotes absolute value)

0026. A fixed set is any sequence of values. Ideally these values should not have any readily discernable relationship or pattern, making hacking the message even more difficult. When a fixed set is used, both the encrypting and the decrypting computer ideally have the fixed set within their own memory. Again, the number of sets which can be used are only limited by the creativity of the developer of such sets. Examples of such sets include:

0027. Set 1 3, 6, 9, 32, 55, 43, 29, 23, 5, 13, 19, 91, 28, 21, 23, 11, 19, 100, 43, 56, 59, 132, 255, 1143, 2329, 623, 65, 613, 919, 91, 128, 421, 823, 711, 19, 0

0028. Set 2, 2, 4, 7, 4, 3, 6, 1, 9, 6, 6, 8, 5, 4

0029. Note, the length of the fixed set isn’t critical to the process as the set can be extended to any required length (to fit the message itself) by simply repeating the fixed set, reversing its order, skipping values when repeating the set, etc. Those of ordinary skill in the art readily recognize a variety of different techniques which allow the fixed set’s length to be extended.

0030. Also note, the values within the key are not limited to a particular range; although some embodiments do limit the values to a set range for ease in computation.

0031. A “key” is possible using historical data. In this method, each new message is used to establish a new “key”. As example, if the message was, “The red dog ran home”, then these values will be used as the key for the second message; and the second message will act as a “key” for the third message; etc.

0032. A “key” can also be made using the computer’s own identification. Such fixed values include the serial numbers of the computers involved and/or the e-mail identifier for the computers. Those of ordinary skill in the art readily recognize a variety of techniques which serve this function. As example, assume the computer’s serial number is: A1JX45812, then a potential initial key is (assigning numerical values to the letters)

\[ 27 36 40 4 5 5 8 1 2 \]

with a subsequent set being defined as the value in the first set added to the next occurring value:

\[ 63 76 44 9 13 9 29 \]

This technique can be repeated as many times as is necessary to provide mapping values for the length of the message being received.

0036. Even further, some “keys” are created using combinations of the above.

0037. For purposes of description, the following are used as the mathematical basis for the preferred embodiment of the invention.

0038. N denotes the number of symbols or characters within the communication;

0039. O_j denotes the original value for the Jth position in the message, J=1, N;

0040. MSG denotes the communication produced by the series O_j, J=1, N;

0041. S_j denotes the adjustment value for the Jth position in the message, J=1, N;

0042. K The key sequence denoted by the series S_j, J=1, N;

0043. E_j denotes the encrypted value for the Jth position in the message, J=1, N. While the preferred embodiment place a range for E_j to fall within the range of O_j, other embodiments do not have this requirement;

0044. M(A,B) denotes the mapping function E_j = M(O_j, S_j), J=1, N, where M is the function that maps the original value O_j using an adjustment value S_j to get the encrypted value E_j;

0045. M'(A,B) denotes a converse map O_j = M'(E_j, S_j), J=1, N which maps the encrypted valued E_j, using the adjustment value S_j to recreate the original message O_j;

0046. X_j denotes the maximum numerical value for O_j; (often this value is fixed for the entire message but in some situations, the maximum value changes during the message);
R(A,B) This function returns the whole number remainder value when A is divided by B (this function is used within the preferred embodiment for the mapping operation).

Using the above references, the preferred embodiment uses a mapping function as indicated:

\[ E_p = \Pi(O_p, S_p) \Rightarrow R(S_r, X_r) \]

Those of ordinary skill in the art readily recognize a variety of other relationships which serve as mapping using the above structures. Examples of these types of mapping are:

\[ E_p = \Pi(O_p, S_p) \Rightarrow R(O_p \times S_p, X_p) \]
\[ E_p = \Pi(O_p, S_p) \Rightarrow R(O_p + S_p, X_p) \]
\[ E_p = \Pi(O_p, S_p) \Rightarrow R(O_p + S_p + X_p, X_p) \]

Further, those of ordinary skill in the art readily recognize alternative mapping functions that are usable in the context described herein.

The invention, to protect a computer from unauthorized programs, has an interface which is configured to load executable programs; these programs are stored in an encrypted form. The interface allows for the withdrawal of and storage of executable programs from a memory mechanism where the executable programs are kept in encrypted form.

During operation, the interface component of the computer system accepts an operator, defined key. This key is used for both the encryption and decryption as outlined above. In the preferred embodiment, the key is collected from the operator. This assures the operator that only he is able to load executable programs onto his computer. Without the proper key, the program will not be decrypted properly and will only be "garbage" and not be able to program the computer.

Note, this embodiment of the invention allows multiple users the ability to use a single computer/computer system without having to share executable programs. Each user is able to decrypt only their own software and retrieve that software using their individual key. In this manner, a large computer system operator is assured that only those that have authority to use a particular program (due to licensing restrictions, security classification, or privacy issues) is provided access to the program.

The encrypted executable program from memory. Using the operator defined key, the interface decrypts the encrypted executable program into a functional executable program and places the functional executable program into the processing unit or its associated working memory.

It is this functional executable program which is used by the processing unit to direct its operation.

During shutdown of the computer, each executable program is checked to see if it was derived from an encrypted executable program; those that aren’t, are verified as being legitimate by the operator prior to their storage into the memory.

In one embodiment, to accomplish the verification, a query is presented to the operator asking if the program should be properly stored (i.e. encrypted before being placed in memory). If the operator consents, the program is considered “authorized” and is encrypted and stored; if the operator does not consent, then the program is “trashed”.

Note, if a “hacker” were to simply place the unauthorized executable program in memory, little or no damage is done. When the computer starts up again and attempts to withdraw the unauthorized program from memory, during the decrypting process, the unauthorized program is scrambled into “garbage”. Little inducement is given for the backer to attempt to plant a worm, spyware, cookie, or “pop-up” program.

A further advantage of the present invention is its ability to check a “key” without having the key accessible to anyone. To accomplish this, the presented key is used to decrypt an encrypted template from the memory into a decrypted template.

The now decrypted template is used as a verifying mechanism to see if the key entered by the operator was properly given or might have been mistyped.

Verifying the decrypted template may be as simple as asking the operator, “Is your name . . .” where the decrypted template is used as the name. Other techniques for verifying the template include a simple check to a unencrypted template or a check to see if the unencrypted template matches the operator provided key. Those of ordinary skill in the art readily recognize a variety of other uses employing the decrypted template.

This technique for checking the key provides a fail-safe method to assure the operator hasn’t mis-typed the key before the key is used in the encryption and decryption process.

The invention, together with various embodiments thereof, will be more fully explained by the accompanying drawings and the following descriptions thereof.

DRA WINGS IN BRIEF

FIG. 1 is a block diagram of the preferred embodiment of the mail server system.

FIG. 2 is a block diagram of the audio/video program download system of the present invention.

FIG. 3 is a block diagram illustrating secure communications between multiple users.

FIG. 4 is a block diagram of an embodiment of the invention used to provide security for a data base.

FIG. 5 is a block diagram showing the use of differing encryption systems between a sender and a receiver.

FIG. 6 is a block diagram of the preferred embodiment for the creation of secure usage of a software program.

FIG. 7 is a flow-chart illustrating an embodiment of the remainder subroutine used in the preferred encryption technique.

FIG. 8 is a flow-chart illustrating the preferred embodiment for the encryption technique.

FIG. 9 is a flow-chart illustrating the preferred embodiment of the decryption technique.
[0073] FIGS. 10A and 10B are flow-charts illustrating an embodiment of the audio/video/program download and play-back respectively.

[0074] FIGS. 11A and 11B are flow-charts of the preferred embodiment's operation for mail for when a message is to be sent and when a message is received.

[0075] FIG. 12 is a table illustrating the preferred embodiment's process.

[0076] FIG. 13 is a block diagram illustrating the invention's protection for executable programs.

[0077] FIG. 14 is a flowchart illustrating the preferred embodiment for the verification of a proper key.

[0078] FIGS. 15A, 15B, and 15C illustrate alternatives for the verification of key input.

[0079] FIG. 16 is a flowchart of the preferred embodiment of the interface used to withdraw encrypted executable programs.

[0080] FIG. 17 is a flowchart of the preferred embodiment of the interface used to store/encrypt executable programs.

**DRAWINGS IN DETAIL**

[0081] FIG. 1 is a block diagram of the preferred embodiment of the mail server system embodiment of the invention.

[0082] Mail server computer 14 is accessible to multiple computers via the Internet 13. For this illustration, three computers are used. Computers 10, 11, and 12, are connected to the Internet 13 and by extension, also to mail server 14.

[0083] Two different types of operations are possible with this configuration:

[0084] (1) one computer wants to communicate with another in a secure manner, but, the two have not done so previously; and,

[0085] (2) two computers wish to securely communicate with each other and have done so previously.

[0086] Addressing the first scenario, computer 10 is equipped with the encryption software (M) and a set of keys as defined above (in an alternative embodiment, computer 10 is configured to establish the key using one of the techniques above); but, computer 11 does not have the decryption software (M') nor any keys.

[0087] The user of computer 10 enters a communication, MSG and a destination address (i.e. the e-mail address for computer 12 or other identifier). Computer 14 determines that this destination has not been used before so one of the keys from the set of keys is arbitrarily selected. Using this key and the mapping function, M. The communication MSG is encrypted.

[0088] The now-encrypted communication, an identifier of the key used, and the destination address, are communicated to the mail server computer 14 by computer 10 via Internet 13.

[0089] Mail server computer 14 recognizes that computer 10 has not previously communicated securely with computer 11. Using the destination information, computer 14 sends an unencrypted message to computer 11 and provides computer 11 with the capability to download the decryption function/software M' together with a single key which is to be used to decrypt the encrypted communication.

[0090] In this manner, computer 11 is provided with the capability to receive secure communications from computer 10; but, computer 11 is not able to send secure communications back to computer 10 (nor to any other computers) without acquiring the encryption mapping capability M together with the entire set of keys.

[0091] In the second scenario, the case where two computers have already established a relationship, computer 10 is equipped with the encryption software (M) and computer 12 has the decryption software (M') together with a set of keys.

[0092] The user of computer 10 enters a communication, MSG and a destination address (i.e. the e-mail address for computer 12 or other identifier). Using the destination address (an identification of computer 12), computer 10 identifies a specific key within the set of keys and uses the specific key with the mapping function M on the communication MSG to create the encrypted message.

[0093] The encrypted message is communicated from computer 10 via the Internet 13 to mail server computer 14. Mail server computer 14, knowing the source of the now-encrypted communication, as well as the destination address (computer 12), determines that these two computers have been in previous secure communications; hence, mail server computer 14 passes the communication along to computer 12.

[0094] In an alternative embodiment, mail server computer 14 decrypts the message from computer 10 and re-encrypts the message specifically for computer 12. This embodiment provides another level of security.

[0095] In yet another embodiment, mail server computer 14 either directly or instructs computer 10 to modify its memory so that the next time a secure communication is sent from computer 10 to computer 12, a different key is used. This modification provides additional security relative to the communications.

[0096] Upon receipt of the encrypted message, computer 12, using the source identifier of computer 10, identifies the proper key from its memory which is to be used in the decryption process. This identified key, together with the decryption mapping function M', allows computer 12 to recreate the original message and display (or place in memory) the original message for the user of computer 12.

[0097] Computer 12 is also able to send a secure communication to computer 10 in a manner as outlined above for a communication between computer 10 and computer 12.

[0098] Note, ideally, the entire encryption/decryption process is "transparent" to the users of computer 11 and computer 12. That is, the users only "see" encrypted material and all encryption and decryption is done automatically.

[0099] FIG. 2 is a block diagram of the audio/video/program download system of the present invention.

[0100] In this situation, the security which is sought isn’t against a third party interloper, but, instead is from the user of computer 22 who, while authorized to obtain the data, may want to download data and then improperly share the
downloaded data with others who have not paid or who are not authorized to have the downloaded data.

[0101] Download server 21 interacts with remote computers via Internet 20. Download server 21 contains digital data which is used to create music, audio, and/or video representations.

[0102] When computer 22 wants to acquire such data, contact is made by computer 22 which requests a specific set of data from download server 21. During the request, computer 22 communicates a key specific to computer 22 which is to be used for the encryption and decryption of the data set. This key is ideally an internally stored value or sequence.

[0103] Using the key for computer 22 and the data, download server 21 encrypts the data and communicates the encrypted data via Internet 20 to computer 22 which stores the encrypted data in memory. While in some embodiments, the data is decrypted prior to storage, in the preferred embodiment of this system, the encrypted data set is stored in memory and is not decrypted until ready for use.

[0104] During use of the encrypted data set by computer 22, portions of the encrypted data set are withdrawn from the memory and are decrypted. This decryption step is accomplished using the internally established key within computer 22; thereby making decryption by any other machine impossible since decryption requires the unique key uniquely found within computer 22.

[0105] To further enhance the security of the downloaded material, ideally, only a portion of the encrypted data set is ever withdrawn and decrypted; without the data ever being fully decrypted, the data is not valuable or usable by any other device except computer 22.

[0106] In like fashion, handheld computer 22 is able to interact with download server 21 via Internet 20 and obtain data which, when used by handheld computer 22 produces music, audio information, or movies.

[0107] FIG. 3 is a block diagram illustrating the secure communications between multiple users.

[0108] In this embodiment, a mail server is not employed, rather, traditional e-mail communications systems are used for the delivery of the messages. Each computer (31, 32, and 33) is able to send messages which have a destination as well as a message (with or without attachments).

[0109] In this embodiment, when a user of computer 31 wants to send a secure transmission to a remote computer 33, computer 31, by knowing the destination, is able to use the appropriate key to encrypt the message and any attachments for computer 33. On receiving the message, since computer 33 knows the source of the message, computer 33 knows the proper key to use in decrypting the message.

[0110] When the user of computer 31 wants to send a secure message to computer 32, a different key is chosen. Computer 31 is creating a series of communications with any number of remote computers, but, each remote computer receives the message in its own unique "language" which is not discernable by the other remote computers. In this manner, unique communications are available. Note, in some situations, a particular key is used with many different computers; but, the selection of the key is still based on the destination computer.

[0111] Should computer 33 receives a message purportedly from computer 31, when the message is decrypted, if the resulting message is gibberish, then computer 33 knows that the message did not originate from computer 31 (since the "language" did not match); conversely, if the message makes sense, then the user of computer 33 is assured of the true source of the message.

[0112] This technique prevents hackers from assuming a false identity merely to gain access to a computer.

[0113] To further enhance this security shield, in one embodiment, a portion of the message being communicated contains an encrypted key which is to be used for the next transmission or reply. This makes it even more difficult for the hacker to counterfeit his identity from the receiving computer. As example, the tenth characters is used as a source in the generation of random numbers by a canned random number generator.

[0114] FIG. 4 is a block diagram of an embodiment of the invention used to provide security for a data base. This embodiment of the invention provides security for a data base which is accessed by many remote sites. Data-base access operations are commonly found in such businesses as: credit card companies; state motor vehicle departments; internal revenue; banking facilities; and many more obvious to those of ordinary skill in the art.

[0115] This embodiment prevents an authorized user of the data base from improperly collecting data from the data base for nefarious uses.

[0116] In this embodiment, data base 45 contains a large amount of proprietary information which is accessible by remote computers 41, 42, and 43. The material within data base 45 is encrypted and remains encrypted using any of the techniques already discussed or others obvious to those ordinary skill in the art.

[0117] When the operator of computer 41 seeks a certain data set, such as that for a particular customer, the inquiry is sent to controller decryption/encryption 44 which identifies the particular data set within data base 45 (which is encrypted) and requests that encrypted information to be sent by data-base 45 to controller/decryption/encryption 44.

[0118] Controller decryption/encryption 44, in the preferred embodiment, decrypts the data set from its stored encrypted state and then re-encrypts the data set using a key which is specific to computer 41. When the second encrypted data set is received by computer 41, computer 41 decrypts the data set for use by the user of computer 41.

[0119] The user of computer 41 is able to manipulate the data set as per their job (such as changing certain elements to reflect such things as an increased loan amount). To store the up-dated data set, computer 41 encrypts the up-dated data set and communicates the encrypted material back to controller 44.

[0120] Controller 44, upon receiving the encrypted data set, recognizes the source of the material and, using the key appropriate for computer 41, decrypts the data set and then re-encrypts the data set commensurate with the encryption technique and key used for data storage within data base 45.

[0121] In this manner, the user of computer 41 is only able to acquire a limited amount of data, as the contents of the data base are kept encrypted using a key which is unknown to the user of computer 41.
FIG. 5 is a block diagram showing the use of differing encryption systems between a sender and a receiver.

As noted earlier, communication between two computers requires that each of the computers is able to identify the source of the information and the address where information is to be sent. This is true whether the transmission is considered an e-mail or an instant message.

As such, computer 51 and computer 52, when communicating with each other via Internet 50, identity themselves and each other with each of the messages being sent. While some embodiments of the invention utilize the same key for the encryption for the outgoing messages (which is also used for the decryption process), in the preferred embodiment each of the computers 51 and 52 use a unique key for the reply message. This causes message 53A to be encrypted differently than message 53B, even though the same two computers are being used for both messages.

This structure keeps someone from being able to re-create the entire “conversation” between computers 51 and 52 without knowing both encryption keys.

This technique is also extremely useful for identifying if the source of the message is who they claim to be, as a hacker will be unable to properly encrypt a message; hence, when the improperly encrypted message is decrypted, “garbage” is created.

FIG. 6 is a block diagram of the preferred embodiment for the creation of secure usage of a software program to prevent the pirating of software.

For explanation of this figure, a software program (such as a spread sheet program) has been stored in the long term memory 63 of the computer. The program within long term memory 63 is encrypted using an identifier (such as the serial number) of the computer as the key for the encryption.

When the program is to be operated, Central Processing Unit (CPU) 60 directs a portion of the program 64A to be withdrawn and decrypted 61. The decrypted portion is communicated to the volatile or working memory (e.g. Random Access Memory RAM, or the like) 62 which is used by CPU 60 in performing the program segment.

When further portions of the program within long term memory 63 are needed, these sections are selectively pulled 64B and 64C, decrypted 61, and used to refresh or replace the contents of RAM 62.

At no time is the entirety of the program within long term memory 63 fully decrypted; rather, only portions of the program are accessible in a decrypted form and hence only a portion of the program is ever available to be “pirated”.

FIG. 7 is a flow-chart illustrating an embodiment of the remainder subroutine used in the preferred encryption technique.

This encryption technique uses a remainder operation in the mapping operations, whether that operation is for encryption or decryption. In this embodiment, the remainder subroutine (R(A,B)) receives the values A and B and returns C, the whole number remainder when A is divided by B.

After the subroutine begins 70A, a pointer is set to zero 71A and the values A and B are obtained 72. A decision is then made if A>B 73A and if so, C is assigned the value A 71B and the subroutine returns C 70B.

If the check of A>B 73A is no, then the pointer is incremented 71C and a determination is made on if the product of P*A>B is made 73B. If the determination is no, then the pointer is incremented again 71C and the process continues until P*A>B (Yes 73B); C is assigned the value of B−(P−1)*A 71D and the program returns the value C 70B.

In this manner, the remainder value is established.

FIG. 8 is a flow-chart illustrating the preferred embodiment for the encryption technique. The mapping function for this encryption is (using the references of above):

\[ E_p = \begin{bmatrix} S_p \times R(S_p, X_p) \end{bmatrix}, \quad X_p = \begin{bmatrix} x_1 \end{bmatrix} \]

Once the program starts 80A, a determination is made to see if the End Of File (EOF) 85 has occurred. An EOF indicates that the entire message has been read. If there has been an EOF, then the program stops 80B; otherwise, the adjustment value from the key (S_p), the maximum number of potential characters (X_p) and the original symbol (O_p) are obtained 81.

The remainder is obtained \( R(O_p, X_p) \) 82A and the value C is returned. The remainder is obtained for \( R(O_p+C, X_p) \) 82B and C is returned. The encrypted value \( E_p \) is assigned the value C and the \( E_p \) is then displayed, communicated, or stored 84. The program then returns to check for the EOF 85.

In this manner, the entire message is encrypted, symbol by symbol using a key for the mapping/encryption process.

FIG. 9 is a flow-chart illustrating the preferred embodiment of the decryption technique.

As noted earlier, ideally the decryption process is performed automatically without any human initiation. In the preferred embodiment of the encryption, the program outlined in FIG. 9 is initiated automatically upon the receipt or opening of an e-mail, instant message, or any other type of message.

Once the program starts 90A, a determination is made on if an End Of File (EOF) has occurred 91A. An EOF indicates that the entire message has been decrypted; hence, on EOF; the program stops 90B.

If there hasn’t been an EOF, then the encrypted letter \( E_p \) is obtained 92A followed by the adjustment value \( S_p \) and the maximum level \( X_p \) 92B. The remainder subroutine is initiated on \( S_p \) and \( X_p \) 93 returning the value C.

A comparison is then made to determine if C is less than the encrypted letter \( E_p \) 91B. If \( C > E_p \), then the original letter \( O_p \) is \( E_p - C \) 94A; otherwise, the original letter \( O_p \) is \( E_p + C - X_p \) 94B.

With the determination of the original letter \( O_p \), the original letter \( O_p \) is displayed (or stored) 95 and the program returns to see if an EOF has now occurred 91A.

In this manner, the entire encrypted message is decrypted letter by letter using the adjustment values as the key and the maximum value to assist in the mapping procedure.
FIGS. 10A and 10B are flow-charts illustrating an embodiment of the audio/video/program download and playback respectively.

Referencing FIG. 10A, the download component, once the program starts 100A, the computer’s identification (i.e. the serial number) is transmitted to the source 101 (where the data is being downloaded from). The source then transmits the encrypted series $E_j$ 102A which is then stored within the computer’s memory 103A. The program then stops 100B.

When the encrypted series $E_j$ is to be played (FIG. 10B), the program starts 100C and an particular value $E_j$ is pulled from memory 102B and this value is decrypted resulting in the decrypted value, the original character/value $O_j$ 104. The original character/value $O_j$ is played 103B.

An EOF check 105 is made. If the EOF has been encountered, then the program stops 100D; otherwise the program loops back and pulls another encrypted value 102B.

FIGS. 11A and 11B are flow-charts of the preferred embodiment’s operation for mail for when a message is to be sent and when a message is received.

A computer, when sending a message (FIG. 11A) starts the program 110A and obtains the destination and message 111A. Using the destination, a key value is determined 112A and the message is encrypted 113A. The encrypted message is then transmitted through normal channels or via a mail server to the destination 112B and the program stops 110B.

An incoming encrypted message is preferably handled as shown in FIG. 11B. The program starts 110C and the source of the message and the encrypted message is obtained 111B. Using the source information, the associated key for decryption is identified 112C and the encrypted message is decrypted 113B. The now-decrypted message is displayed for the user 114 and the program stops 110D.

FIG. 12 is a table illustrating the preferred encryption and decryption process.

Using the preferred mapping function $(E_i, M(O_j, S_j) = [R(S_j + R(S_j, X_i))])$, FIG. 12 illustrates how the message: “the red dog ran home” 120 is first encrypted and then decrypted.

For this example, the numerical values range from 0=blank space, 1=“a”, 2=“b” . . . 25=“y”, and $X_j$ is a constant value 26.

In this example, the key $S_j$ 121 which is used is defined by the series:

| 4 20 6 21 22 39 27 48 4 14 32 7 81 0 17 17 14 |
| 42 8 4 |

As illustrated, the receiving computer (doing the decryption) uses a reversing algorithm together with the key set $S_j$, which were also used in the encryption operation.

The power of this particular encryption technique is clear when the original message is compared to the encrypted message which is communicated over the distributed network of computers.

Original Message: the red dog ran home 121
Transmitted Message: xbkunrevhcmguaeqveni 122

thereby providing encryption which is unique between the two parties and making the transmission difficult if not impossible to decrypt.

FIG. 13 is a block diagram illustrating the invention’s protection for executable programs.

Processing unit 130 receives its operating programs, such as executable programs, from the interface 131 which serves to decrypt the operating programs held in memory 132. The operating programs are placed within operating memory 133 by the processing unit and are withdrawn as needed.

In like fashion, when processing unit 130 shuts down, a check is made of the programs it has and a match is made to the memory unit, any programs not in memory are communicated to operator 136 via monitor 134 and operator 136 responds if these programs should be kept or “trashed”. Operator 136 responds via keyboard 135 or some other type of input device such as a mouse or voice recognition microphone.

In this way, only the programs that the operator wants to keep are encrypted for later use while the improper or hacked programs are discarded.

In this diagram, memory 132, interface 131, processing unit 130 and operating memory 133 are shown as distinct boxes, but, in alternative embodiments, these functional operations are contained within a single housing.

FIG. 14 is a flowchart illustrating the preferred embodiment for the verification of a proper key.

Once this program or subroutine starts 140A, the proposed key is obtained from the operator. Using the proposed key, the encrypted template is decrypted 142. The proposed key is then verified 143. If the proposed key is verified (Yes), then the program/subroutine stops 140B; otherwise (the proposed key is not verified, a NO), the operator is informed that the proposed key is not the valid key 144 and another proposed key is obtained 141.

In this manner, should the operator enter an improper key, a simple check reveals it as improper prior to any encryption being done. Also, since the proposed key is used to decrypt a template for comparison, and since this template is encrypted, a hacker is unable to obtain access to the key by getting into the unencrypted material within the computer; hence, a further level of security is added by this technique.

FIGS. 15A, 15B, and 15C illustrate alternatives for the verification of key input as first discussed relative to FIG. 14.

Referencing FIG. 15A, verification of the proposed key is accomplished in this embodiment by first displaying the decrypted template to the operator 150 and obtaining an authorization indicia from the operator 151.

As example, the template may be as simple as the operator’s name. Once the operator’s name is decrypted, it is shown to the operator who responds with a “YES” or a “NO” depending on if the display shows his name correctly.
This embodiment then branches 152A based upon the authorization response (element 143 of FIG. 14).

FIG. 15B shows another technique for verification in which the proposed key is used to decrypt the template which is also the proper key.

As example, assume the proper key is: “September 7, 1952”; the proper key has previously been encrypted using itself for the encrypting process and the encrypted key is placed in the memory.

Later, when an operator enters a proposed key, the proposed key is used to decrypt the encrypted proper key. Should the proposed key not be correct, then the decryption will not be performed properly and the decrypted proper key will not match the proposed key.

This type of sorting is done through a simple logical function which compares the template (the proper key) with the proposed key; if they are identical then encryption can proceed; if not, then, as outlined in FIG. 14 (element 143), the operator is informed and another proposed key is obtained.

FIG. 15C illustrates yet another technique for ascertaining that the proposed key is proper prior to any encryption operation.

In this embodiment, the operator is queried as the substance of a comparison template 153. This comparison template is used to compare with the decrypted template 154 for a verification operation.

One operation for this embodiment would be where the operator provides not only a proposed key, but also the template. In this case, the operator would provide a proposed key, such as “my name is” and a comparison template, “Mark Oram”. The program uses the proposed key “my name is” to decrypt the template within memory and if the decrypted template doesn’t match “Mark Oram”, then the proposed key isn’t correct.

In all of these embodiments, a level of security is provided that prevents any encryption of programs or data using an improper key.

FIG. 16 is a flowchart of the preferred embodiment of the interface used to withdraw encrypted executable programs.

Once the program/subroutine starts 160A the identified program is withdrawn from memory in its encrypted form 161A. The encrypted form is decrypted 161B as outlined above and is sent to the processing unit 161C (which stores the decrypted program in the operating memory). The program then stops 160B.

FIG. 17 is a flowchart of the preferred embodiment of the interface used to store/encrypt executable programs.

At the start of the program/subroutine 170A, the executable program within the operating memory is identified 171A. A comparison is then made to determine if that executable program has a counterpart within the encrypted programs 172A.

If there is a corresponding encrypted program (YES) then the executable program is removed from the operating memory 171E and a determination is made on if there are any more executable programs within the operating memory to consider 172D. If there are no more executable programs to consider the program stops 170B; otherwise, the next executable program within the operating memory is withdrawn 171A for consideration.

Should there be no corresponding encrypted program (172A, “NO”), then a determination is made on if encryption key is needed 172B. If the encryption key has not been previously provided (YES) then the key is obtained from the operator 173 and the key is verified 171B as outlined above.

Once the key has been obtained, a determination is made on if the program should be stored 172C. This decision is accomplished typically by presenting the program to the operator to accept/reject; in other embodiments, only selected programs can be added and these programs are defined within the computer files.

If the decision is not to store the executable (”NO”) 172C, then the executable is removed and the program continues. If the decision is to store the executable (”YES”) 172C, then the program is encrypted 171C using the key previously obtained and the encrypted program is stored 171D in memory and the executable program is removed 171E as outlined above.

In this manner, any and all of the executable programs are scanned and only the proper ones are stored in encrypted form.

Note, during an uncontrolled shut down of the computer (such as power outage), only the previously encrypted programs are kept.

Using the technique above, the present invention provides for a highly secure structure for preventing unauthorized executable programs from being stored within a computer.

What is claimed is:

1. A computer system comprising:
   a) a memory;
   b) a processing unit being configurable by executable commands; and,
   c) an interface configured to,
      1) withdraw an encrypted executable program from said memory,
      2) decrypt said encrypted executable program into a functional executable program, and,
      3) communicate said functional executable program to said processing unit.

2. The computer system according to claim 1, wherein said interface is further configured to:
   a) identify an executable program within said processing unit which does not have a encrypted counterpart within said memory;
   b) query an operator of said computer system if said executable program is to be saved and receive an authorization response from said operator; and,
   c) based upon said authorization response, selectively,
      1) encrypt said executable program into an encrypted executable program, and,
2) store said encrypted executable program within said memory.

3. The computer system according to claim 2, wherein, said interface is further configured to remove said executable program from the memory and the processing unit.

4. The computer system according to claim 2,
   a) wherein said interface is further configured to accept a key from said operator; and,
   b) wherein said key is used when said interface operates to encrypt said executable program into an encrypted executable program.

5. The computer system according to claim 4, wherein said interface is further configured to verify said key as being accurate.

6. The computer system according to claim 5, wherein, during operation of said interface to verify said key as being accurate, said interface is further configured to:
   a) using said key, decrypt an encrypted template from the memory into a decrypted template; and,
   b) verify the decrypted template as being accurate.

7. The computer system according to claim 6, wherein, during operation of said interface to verify the decrypted template as being accurate, said interface is configured to:
   a) present the decrypted template to the operator of said computer;
   b) receive an acceptance response from the operator of said computer; and,
   c) based upon said acceptance response, permit said interface to encrypt said executable program into an encrypted executable program.

8. The computer system according to claim 6, wherein, during operation of said interface to verify the decrypted template as being accurate, said interface is configured to:
   a) compare said decrypted template to said key and generate a comparison indicia therefrom; and,
   b) based upon said comparison indicia, permit said interface to encrypt said executable program into an encrypted executable program.

9. The computer system according to claim 6, wherein, during operation of said interface to verify the decrypted template as being accurate, said interface is configured to:
   a) compare said decrypted template to a predefined template and generate an comparison indicia therefrom; and,
   b) based upon said comparison indicia, permit said interface to encrypt said executable program into an encrypted executable program.

10. A loading system configured to:
   a) decrypt an encrypted executable program into a functional executable program; and,
   b) communicate said functional executable program to a processing unit.

11. The loading system according to claim 10, further configured to:
   a) identify an executable program within said processing unit which does not have a encrypted counterpart;
   b) query an operator if said executable program is to be saved;
   c) receive an authorization response from said operator; and,
   d) based upon said authorization response, encrypt said executable program into an encrypted executable program.

12. The loading system according to claim 11, further configured to store said encrypted executable program within a memory.

13. The computer system according to claim 11, wherein an operator defined key is used when said interface operates to encrypt said executable program into an encrypted executable program.

14. The computer system according to claim 13, wherein said interface is further configured to:
   a) using said operator defined key, decrypt an encrypted template into a decrypted template; and,
   b) verify the decrypted template as being accurate.

15. A computer system comprising:
   a) a memory;
   b) a processing unit being configurable by executable commands; and,
   c) a loading interface configured to,
      1) accept an operator defined key,
      2) withdraw an encrypted executable program from said memory,
      3) using said operator defined key, decrypt said encrypted executable program into a functional executable program, and,
      4) communicate said functional executable program to said processing unit; and,
      5) wherein said processing unit operates said functional executable program.

16. The computer system according to claim 15, wherein said loading interface is further configured to:
   a) identify an executable program within said processing unit which does not have a encrypted counterpart within said memory;
   b) query an operator of said computer system if said executable program is to be saved and receive an authorization response from said operator; and,
   c) based upon said authorization response, selectivity,
      1) using said operator defined key, encrypt said executable program into a second encrypted executable program, and,
      2) store the second encrypted executable program in said memory.

17. The computer system according to claim 16, wherein, said interface is further configured to, based on said authorization response, remove said executable program from the memory and the processing unit.

18. The computer system according to claim 16, wherein said downloading interface is further configured:
a) Using said key, decrypt an encrypted template from memory into a decrypted template;

b) present the decrypted template to the operator of said computer;

c) receive an acceptance response from the operator of said computer; and,

d) based upon said acceptance response, permit said interface to encrypt said executable program into an encrypted executable program to verify said key as being accurate.

* * * * *