

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

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
15 November 2001 (15.11.2001)

PCT

(10) International Publication Number
WO 01/86392 A2

(51) International Patent Classification⁷: **G06F 1/00**

(21) International Application Number: PCT/US01/14807

(22) International Filing Date: 8 May 2001 (08.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/567,886 10 May 2000 (10.05.2000) US

(71) Applicant: **THE PROCTER & GAMBLE COMPANY**
[US/US]; One Procter & Gamble Plaza, Cincinnati, OH
45202 (US).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AT (utility model), AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, CZ (utility model), DE, DE (utility model), DK, DK (utility model), DM, DZ, EE, EE (utility model), ES, FI, FI (utility model), GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (utility model), SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, 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, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— without international search report and to be republished upon receipt of that report

(72) Inventor: **JOHNSTON, Stephen, Edward**; 2573 Westpoint Court, Burlington, KY 41005 (US).

(74) Agents: **REED, T., David** et al.; The Procter & Gamble Company, 5299 Spring Grove Avenue, Cincinnati, OH 45217-1087 (US).

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND APPARATUS FOR CENTRALIZED AUTHENTICATION

(57) Abstract: A method for centralized user authentication on a computer network where each of a plurality of agent computers perform the method. A user identifier and token is obtained by at least one of the plurality of networked computers. The user identifier and token are concatenated and passed to a central authentication system on the network. A response is received from the central authentication system, and if the response does not permit access, the token is passed as a password to the central authentication system. Other systems and methods for centralized authentication are also disclosed.



WO 01/86392 A2

METHOD AND APPARATUS FOR CENTRALIZED AUTHENTICATION

A portion of the disclosure of this patent document contains material which is
5 subject to copyright protection. The copyright owner has no objection to the facsimile
reproduction by anyone of the patent document or the patent disclosure, as it appears in
the Patent and Trademark Office patent file or records, but otherwise reserves all
copyright rights whatsoever.

Technical Field

10 The present invention relates generally to authentication in computer systems, and
will be specifically disclosed as a method and apparatus for centralized user
authentication in a computer network.

Background of the Invention

15 The virtual explosion of technical advances in microelectronics, digital computers
and software have changed the face of modern society. In fact, these technological
advances have become so important and pervasive that this explosion is sometimes
referred to as "the information revolution." Through telephone lines, cables, satellite
20 communications and the like, information and resources are ever increasingly being
accessed and shared.

To prevent unauthorized access to programs and/or data, computer systems often
use various security measures, two examples of which include physical security and login
security. As the name implies, physical security refers to limiting physical access to a
25 given computer resource. For instance, servers in a client/server network are often
maintained in a locked room with limited access. Login security can vary greatly from
one computer system to the next. One common form of login security comprises a login
phase and an authentication phase. The login phase typically involves prompting a
source (such as a user, a program, a resource, etc.) which is attempting to enter the
30 system for a name and a password. After successfully proving knowledge of the
password, the source is authenticated and is allowed whatever level of access is permitted
for the source.

Beyond variations in the type of login security, the content of login security often
varies from one computer system to the next. For example, a user may have a user name
35 and password for one system and have a different user name and password for another
system. Accordingly, the user in the previous example is required to remember his user

name and password for both systems. The problem is even further compounded in a network of computers having a plurality of computers and programs each operating its own login security system, wherein the user may potentially be required to remember a user name a password for each of the different computers and programs.

5

Summary of the Invention

Accordingly, an object of the invention is to provide an improved method and apparatus for authentication in a computer system. Additional objectives, advantages and novel features of the invention will be set forth in the description that follows and, in part, will become apparent to those skilled in the art upon examining or practicing the invention. The objects and advantages of the invention may be realized and obtained by means of the instrumentalities and combinations particularly pointed out in the appended claims.

One aspect of the present invention is a computer system comprising a network of computers. An authentication server on the network comprises an authentication engine having a first state wherein the authentication engine uses a password comprising two elements and a second state wherein the authentication engine uses a password comprising one element. The authentication engine further comprises an application programming interface through which other computers of the network may make requests to the authentication engine. A plurality of agent computers on the network each comprise agent software for the authentication server being encoded for sending a first authentication request via the application programming interface using a password comprising two elements, and if authentication is not established in response to the first authentication request, sending a second authentication request via the application programming interface using a password comprising one element.

Another aspect of the invention is a method for centralized user authentication on a computer network where each of a plurality of agent computers perform the method. A user identifier and token is obtained by at least one of the plurality of networked computers. The user identifier and token are concatenated and passed to a central authentication system on the network. A response is received from the central authentication system, and if the response does not permit access, the token is passed as a password to the central authentication system.

Yet another aspect of the invention is a method for centralized user authentication on a computer network. A user identifier and token are obtained by an agent computer of a central authentication system. The user identifier and token are passed to the central

authentication system on the network. If the central authentication system requires the user identifier and token, the passed user identifier and token are processed by the central authentication system to determine whether the user is authenticated. If the central authentication system requires only the token, the passed token is processed by the central authentication system to determine whether the user is authenticated.

Still other aspects of the present invention will become apparent to those skilled in the art from the following description of a preferred embodiment, which is by way of illustration, one of the best modes contemplated for carrying out the invention. As will be realized, the invention is capable of other different and obvious aspects, all without departing from the invention. Accordingly, the drawings and descriptions are illustrative in nature and not restrictive.

Brief Description of the Drawings The accompanying drawings, incorporated in and forming part of the specification, illustrate several aspects of the present invention and, together with their descriptions, serve to explain the principles of the invention. In the drawings:

Fig. 1 illustrates an example of a computer;

Fig. 2 illustrates an example of a computer network;

Fig. 3 illustrates an agent and authentication engine;

Fig. 4 depicts a flowchart of an authentication process;

Fig. 5 depicts a flowchart of a next token process;

Fig. 6 depicts a flowchart of a new PIN mode process;

Fig. 7 depicts a flowchart to process a client request; and

Fig. 8 depicts a dialog box;

Fig. 9 depicts a centralized authentication system used for a variety of different types of transactions; and

Figs. 10-18 depict illustrations of exemplary dialog boxes of the present invention.

Detailed Description

Reference will now be made to the present preferred embodiment of the invention, an example of which is illustrated in the accompanying drawings, wherein like numerals indicate the same element throughout the views.

Often computers communicate between each other and share information, applications and/or services. Computers or computer systems are generally any device capable of processing data in accordance with one or more instructions. Examples of computers include personal computers, workstations, servers, mainframes, embedded systems, microprocessors, discrete logic systems, analog systems, and the like. Sometimes in the setting of interconnected computers, the various computers are referred to as nodes, which is a generic term referring to a point in a interconnected system.

Turning now to Fig. 1, the present invention may be implemented, for example, by operating a computer system 1000 to execute a sequence of machine-readable instructions. These instructions may reside in various types of signal bearing media, such hard disk drive 1010 and main memory 1020. In this respect, another aspect of the present invention concerns a program product, comprising signal bearing media embodying a program of machine-readable instructions, executable by a digital data processor, such as central processing unit (CPU) 1030, to perform method steps. The machine-readable instructions may comprise any one of a number of programming languages known in the art (e.g., C, C++, etc.).

It should be understood that the present invention may be implemented on any type of computer system. Accordingly, the present invention is not limited to implementation on the type computer system shown in Fig. 1. As shown, computer system 1000 comprises main or central processing unit (CPU) 1030, which is connected to main memory 1020 (e.g., random access memory (RAM)), a display adapter 1040, an auxiliary storage adapter 1050, and a network adapter 1060. These system components are interconnected through the use of a system bus 1070.

CPU 1030 may be, for example, a Pentium Processor made by Intel Corporation. However, it should be understood that the present invention is not limited to any one make of processor and the invention may be practiced using some other type of a processor such as a co-processor or an auxiliary processor. Auxiliary storage adapter 1050 is used to connect mass storage devices (such as hard disk drive 1010) to computer system 1000. The program need not necessarily all simultaneously reside on computer system 1000. Indeed, this latter scenario would likely be the case if computer system 1000 were a network computer, and therefore, be dependent upon an on-demand shipping mechanism for access to mechanisms or portions of mechanisms that resided on a server. Display adapter 1050 is used to directly connect a display device to computer system 1000. Network adapter 1060 is used to connect computer system 1000 to other computer systems.

It is important to note that while the present invention has been described in the context of a fully functional computer system, those skilled in the art will appreciate that the mechanisms of the present invention are capable of being distributed as a program product in a variety of forms, and that the present invention applies equally regardless of the particular type of signal bearing media used to actually carry out the distribution. Examples of signal bearing media include: recordable type media, such as floppy disks, hard disk drives, and CD ROMs and transmission type media, such as digital and analog communications links and wireless.

Fig. 2 illustrates an example of a network 10, shown here as a client/server network implemented as a local area network "LAN" or wide area network "WAN". As one with ordinary skill in the art will readily appreciate, a client/server network is only one type of network, and a variety of other configurations, such as local area networks, wide area networks, peer-to-peer connections, modem connections, the Internet, and the like, are also considered networks. In a client/server network, a plurality of nodes are interconnected such that the various nodes send and/or receive information to/from a server and one another. As shown here, a plurality of server nodes 11 are interconnected to a plurality of client nodes 12 using a communication link 13, such as network interface cards (e.g., Ethernet, token ring, etc.), parallel cables, serial cables, telephone lines, universal serial bus "USB", Firewire, IEEE-1394, Bluetooth, fiber optics, infrared "IR", radio frequency "RF", and the like.

Fig. 2 also depicts a computer readable medium 14, shown here as a floppy diskette. A computer readable medium stores information readable by a computer, such as programs, data files, etc. As one with ordinary skill in the art will readily appreciate, a computer readable medium can take a variety of forms, including magnetic storage (such as hard drives, floppy diskettes, tape, etc.), optical storage (such as laser disks, compact disks, digital video disks "DVD", etc.), electronic storage (such as random access memory "RAM", read only memory "ROM", programmable read only memory "PROM", flash memory, memory sticks, etc.), and the like. Some types of computer readable media, which are sometimes described as being non-volatile, can retain data in the absence of power so that the information is available when power is restored.

The network 10 includes a central authentication system 15. The central authentication system 15 is a centrally managed point to authenticate users on the network 10 and provides a centralized authentication source for the network 10. The central authentication system 15 can take a variety of different forms including a single machine, a distributed system, a master/slave arrangement, and the like. Preferably, the

central authentication system 15 uses a strong two-factor authentication system. As shown in the present embodiment, the central authentication system 15 takes the form of an authentication server. For example, the authentication server may comprise Access Control Encryption "ACE" server from RSA Security, Inc. The other computers on the network 10, including clients 12 or other servers 11, can be authentication agents of the authentication server 15. Sometimes an authentication agent is referred to as a client of the authentication server or an agent machine. An agent machine is a known and trusted entity to the authentication server 15 and is registered with the authentication server 15. As shown in Fig. 3, the agent machine 20 which is an authentication agent of the authentication server 15, comprises an agent 21, which typically takes the form of executable software running on the agent machine 20. The agent 21 is adapted to send authentication requests via the application program interface "API" 22 for the authentication engine 24 operating on the authentication server 15. The authentication engine 24 processes authentication requests and replies via the API 22 whether or not a user have been authenticated. As one skilled in the art will appreciate, the method of the present invention may be performed directly by the authentication agent 21, the API 22, or potentially in the authentication engine 24. Preferably, the authentication agent 21 is adapted to send authentication requests via API 22 for the authentication engine 24 operating on the authentication server 15.

An authentication server 15 uses a strong two-factor authentication engine 24. Authentication requests through the API 22 comprise two variables: a login identification and a password (sometimes the password is referred to as a passcode). The login identification, often referred to as login ID, typically takes the form of an alpha-numeric string unique to a user. For instance, the login ID for Steve Johnston may be "SJohnston." The login ID is registered with the authentication engine 24 and is known by the user.

If a strong authentication system is used, the password may comprise a PIN in combination with a token or biometric reading. The biometric reading may include fingerprint, voiceprint, retina scan, thermal imaging or the like. The content of the password depends upon the state of the authentication engine 24 for the account associated with the login ID. For instance, the authentication server may include three states or modes: normal, new PIN, and next token. In normal mode, the password is a combination of two values concatenated with one another: a personal identifier and a token. The personal identifier is an alpha-numeric string, such as a personal identification number "PIN." The token is also a alpha-numeric string, which is

preferably generated by a token generator, such as the SecurID authentication provided by RSA Security Inc. As one skilled in the art will readily recognize, the SecurID system is a parallel system for generating tokens. A process operating with the user periodically generates a numeric code. A similar process operating with the authentication engine 24
5 synchronously generates the same numeric code as the process running with the user. Accordingly, at any time the user and the authentication engine 24 both have common knowledge of a token. New PIN mode occurs when the authentication engine 24 does not recognize the PIN associated with the user, such as for a first time user, when the user forgets his PIN and an administrator resets the mode to "New PIN" mode, a threshold
10 number of failures occurs, or the like. When the user is in new PIN mode, the password passed by the API 22 to the authentication engine 24 consists only of the token. Next token mode can be triggered by electronic drift in the parallel to the generator, a threshold number of login failures followed by a success login, and the like. When a user is in next token mode, the password passed by the API 22 to the authentication engine 24 is the
15 same as normal mode having a concatenated value of the personal identifier and token. While the values passed by the API 22 to the authentication engine 24 varies on the state of the user, only the authentication engine 24 has knowledge of the state, so the agent 21 will not know what information needs to be passed for the password.

Fig. 4 depicts a flowchart of an authentication process 30, portions of which are
20 preferably implemented as a program running with or part of the agent 21. For instance, the process logic can be compiled integrally with the agent 21. Alternatively, the process logic can be implemented as a function, such as a dynamic link library "DLL" or the like triggered by the agent 21. At step 31, the agent 21 prompts the user to enter his login ID, PIN, and token. Preferably, the user is presented a dialog box with separate fields for
25 each value. At step 32, an API request is prepared by the agent 21 comprising the login ID and password concatenated with the PIN and token. The authentication request is sent by the API 22 to the authentication engine 24. When the authentication engine 24 responds, the agent at step 33 determines whether the authentication was valid. If valid, at step 34 the process determines whether the user is in next token mode. If so, the next
30 token process 40 is implemented, and if not a success acknowledgment "ACK" is sent to the agent 21. If the authentication was invalid, the agent prepares a new API request comprising the login ID and password using only the token. At step 35 if the new request is valid, the new PIN mode process 50 is implemented. If, however, the authentication request was invalid, the process determines at step 36 whether the account for the entered

login ID was disabled. If disabled, an account disabled ACK is sent to the agent 21, and if not disabled a try again ACK is sent to the agent 21.

In another embodiment, the agent 21 prepares the API request by mixing the PIN and token together to form a Passcode. This mixing can be based on a pre-defined algorithm. The authentication request is sent by the API 22 to the authentication engine 24. The authentication engine 24 contains the same algorithm to authenticate the mixed PIN and token (Passcode). In yet another embodiment, the authentication engine 24 may utilize the algorithm to “un-mix” the Passcode to obtain the PIN and token.

In another embodiment, portions of the process logic 30 can be incorporated with the authentication engine 24. For instance, a concatenated password of the PIN and token could be passed to the authentication engine 23, and at step 35 the authentication engine 24 would parse the password to read only the token, preferably by extracting only the last number of characters in the password corresponding to the length of the token. In another embodiment, the authentication engine 24 at step 35 calculates the token from the password using a reverse hash algorithm. For example, a user enters a Login ID and token generated using a token generation device with hashing for an entered PIN. A user inputs the PIN in the token generation device and the device generates a specific token from an algorithm incorporating the PIN. The user enters the hashed token and the Login ID, wherein the authentication agent passes the token and Login ID to the authentication engine 24. The authentication engine 24 deconstructs the hashed token to verify the token by utilization of a stored PIN corresponding to the Login ID.

Alternatively, the API 22 could accept separate variables for the login ID, user identifier, and token, as opposed to a concatenated password. Accordingly, the authentication engine 24 need only read the variable for the token.

Fig. 5 depicts a flowchart for a next token mode process 40. At step 41, a next token request is sent to the agent 21. The agent 21 at step 42 prompts the user with the previous token and asks the user to wait till the next token is displayed on the SecurID. At step 43, the user enters the next token, which is passed to the authentication engine 24 from the authentication agent 21 or its API 22. During step 44, the next token is evaluated to determine if it is valid. If so, at step 45 a success message is sent to the agent 21 followed by a success ACK. If not, the process determines at step 46 whether a threshold number of attempts have been reached. If not, the process returns to step 41 and the number of attempts is incremented. If so, a failure message is sent to the agent 21 at step 47 followed by a failure ACK.

Fig. 6 depicts a flowchart for a new PIN mode process 50. The new PIN mode is an administratively set mode, i.e. an administrator must configure the user's authentication state to new PIN mode. At step 51, a new PIN request is sent to the agent 21 from the authentication engine 24. The agent 21 at step 52 prompts the user to enter and reconfirm the previously entered PIN. At step 53, the user enters the reconfirmation PIN. During step 54, the reconfirmation PIN is compared to the previously entered PIN by the agent 21 to see if they match. If so, at step 55, the PIN data is sent to the authentication engine 24 by the agent 21. The authentication engine sets the PIN for the user's account at step 57. Then, at step 58, the authentication engine sends a success acknowledgment to the agent for the final process. If the reconfirmation PIN does not match the previously entered PIN at step 54, the client prompts the user to enter and verify a new PIN (step 56). The user enters and reconfirms the PIN at step 59. The process then repeats step 54 to determine if the PIN numbers match.

Fig. 7 depicts a flowchart of agent process 60 based on ACK messages. ACK messages are received and processed at block 61. If a try again ACK was received, the process moves to the authentication process 30. If a success ACK was received, at block 62 authentication is a success and the user is allowed to proceed. If a failure ACK was received, at block 63 the user is presented a authentication failed message and the user is not allowed access. If an account disable ACK was received, the user is presented at block 64 with an account disable message, followed by an authentication failed message.

Consider the following MFC example programmed in Microsoft Visual C++. In a typical embodiment, a dialog box is presented prompting the user for user name and a passcode. In the present example, a dialog box is presented in the form of Fig. 8.

The above code grabs the PIN value entered in the dialog box of Fig. 8 and sets it in a variable strPassword. When the OK button on the dialog box is selected, the Class CdlgSma::OnOK() listed below is invoked.

```

5
  BOOL CDlgSma::SmaLogonUser(CString strUserid, CString strPassword, CString
  strToken, int& connState)
  {
      BOOL bRet = FALSE;

10      bRet = m_smaAuth.Logon(strUserid, strPassword, strToken, connState);

      // if bRet == TRUE, connState has been filled in by the m_smaAuth.Logon()
      return bRet;

15  }

  BOOL CSmaAuth::Logon(CString strUserid, CString strPassword, CString strToken,
  int& connState)
20  {
      OutputDebugString("--> Entering int CSmaAuth::Logon()\n");

      #ifdef _DEBUG
      CString str;
      str = "Inside CSmaAuth::Logon(";
25      str += "strUserid = " + strUserid + ", ";
      str += "strPassword = " + strPassword + ", ";
      str += "strToken = " + strToken + """;
      OutputDebugString(str);
30      #endif // _DEBUG

      int retcode = 0;

35      retcode = AceSetUsername(m_hSdi, (LPSTR)(LPCSTR)strUserid);

```

```
        if (retcode != ACE_SUCCESS) {
            m_ErrorCode = retcode;
            m_strError.Format("Error during Authentication (AceSetUsername).\n"
                "Error code = %d", m_ErrorCode);
5
            OutputDebugString("<== Exiting int CSmaAuth::Logon()\n");
            return FALSE;
        }

10        retcode = AceSetPasscode(m_hSdi, (LPSTR)(LPCSTR)(strPassword +
strToken));
        if (retcode != ACE_SUCCESS) {
            m_ErrorCode = retcode;
            m_strError.Format("Error during Authentication (AceSetPasscode).\n"
15                "Error code = %d", m_ErrorCode);

            OutputDebugString("<== Exiting int CSmaAuth::Logon()\n");
            return FALSE;
        }

20        m_pSmaEvent->SetCaller(LOGON);
        retcode = AceCheck(m_hSdi, SmaCallBack);
        if (retcode != ACE_PROCESSING) {
            m_ErrorCode = retcode;
25            m_strError.Format("Error during Authentication (AceCheck).\n"
                "Error code = %d", m_ErrorCode);

            OutputDebugString("<== Exiting int CSmaAuth::Logon()\n");
            return FALSE;
30        }

        retcode = m_pSmaEvent->Wait(INFINITE);
        if (retcode != WAIT_OBJECT_0)
        {
35            m_ErrorCode = retcode;
```

```
m_strError.Format("Error during Authentication (SmaWaitEvent).\n"
    "Error code = %d", m_ErrorCode);

OutputDebugString("<== Exiting int CSmaAuth::Logon()\n");
5    return FALSE;
}

retcode = m_pSmaEvent->GetConnState(connState);
if(retcode != ACE_SUCCESS)
10    {
        // Error occurred during the processing of AceCheck, signalled by
        SmaCallback
        m_ErrorCode = retcode;
        m_strError.Format("Error during Authentication (GetConnState).\n"
15            "Error code = %d", m_ErrorCode);

        // Do not AceClose, because this error may be Authentication failure
        // (or something else) error
        OutputDebugString("<== Exiting int CSmaAuth::Logon()\n");
20        return FALSE;
    }

    // AceCheckCompleted successfully,
    // The connState can be ACM_OK, ACM_NEXT_CODE_REQUIRED,
25    ACM_NEW_PIN_REQUIRED etc

    OutputDebugString("<-- Exiting int CSmaAuth::Logon()\n");
    return TRUE;
} // end of CSmaAuth::Logon()
30

void CDlgSma::OnOK()
{
    int nId = 0;
35    // TODO: Add extra validation here
```

```

switch(m_dwState)
{
case LOGON:
    // login on, normal case
5      {
        static nTriesLogon = 0;

        BOOL bRet = FALSE;
10      int connState = -1;
        CString strUserid, strPassword, strToken;
        strUserid = GetUserid();
        strPassword = GetPassword();
        strToken = GetToken();
15      if((nId = ValidateLogonEntries(strUserid, strPassword, strToken))
        != 0)
        {
            // invalid entries, return
            GotoDlgCtrl(GetDlgItem(nId));
20          return;
        }

        // Store the password for later verification in case of New pin mode
        m_strUsername = strUserid;
25      SetUsernameinReg(m_strUsername); // set the user name in
        registry

        m_strPassword1 = strPassword;
        m_strToken = strToken;

30      StartAnimation();
        (A) bRet = SmaLogonUser(strUserid, strPassword, strToken,
        connState);

        StopAnimation();
        if(bRet)
35      {

```

```

        if(connState == ACM_OK)
        {
            // connected
            TerminateDialog(CONNECTED);
5           return;
        } // end of if(connState == ACM_OK)

        if(connState == ACM_NEXT_CODE_REQUIRED)
        {
10           // next token mode (Resync)
            SetDlgType(RESYNC);
            return; // do not terminate dialog, we are in a new
mode
        } // end of if(connState ==
15 ACM_NEXT_CODE_REQUIRED)

        if(connState == ACM_ACCESS_DENIED)
        {
            // we may be in new pin mode, lets try by sending
20 Userid and token (no pin)
            (B) bRet = SmaLogonUser(strUserid, "", strToken,
connState);

            if(bRet)
            {
25
                if(connState ==
ACM_NEW_PIN_REQUIRED)
                {
                    // We are in new pin mode
30 SetDlgType(NEWPIN);
                    return;
                }
                else
                {
35 // Access is denied

```

```

                                                                    AfxMessageBox("Incorrect userid and/or
password. Please retry.");
                                                                    SetDlgType(RETRYLOGON);
                                                                    return;
5                                                                    }
                                                                    }
                                                                    else
                                                                    {
                                                                    // Error in 2nd execution of SmaLogonUser,
10 cannot continue
                                                                    AfxMessageBox(m_smaAuth.m_strError);
                                                                    TerminateDialog(DISCONNECTED);
                                                                    return;
                                                                    }
15 } // end of if(connState == ACM_ACCESS_DENIED)
                                                                    else
                                                                    {
                                                                    // if we come here, it is because of an unexpected
response
                                                                    CString strError;
20 strError.Format("Unexpected connection status
received. "
                                                                    "Please restart application. [connState =
%d", connState);
25 AfxMessageBox(strError);
                                                                    TerminateDialog(DISCONNECTED);
                                                                    return;
                                                                    }
                                                                    } // end of if(bRet), for the 1st call the SmaLogonUser
30 else
                                                                    {
                                                                    // Error in 1st execution of SmaLogonUser, cannot continue
                                                                    AfxMessageBox(m_smaAuth.m_strError);
                                                                    TerminateDialog(DISCONNECTED);
35 return;
```



```
        }  
    } // end of LOGON block  
    break;
```

5

In the foregoing code, an authentication request (*A*) is executed where both the strPassword (i.e. PIN) and strToken are passed to the function SmaLogonUser() which invokes the class CsmaAuth::Logon(). If the authentication API responds ACM_ACCESS_DENIED, typical implementations would present a message box denying access and the process ends. In the above code another attempt (*B*) is executed where the authentication request is made without the strPassword variable (i.e. using the token only). If the authentication API responds ACM_NEW_PIN_REQUIRED, the normal logic for new PIN mode is employed. If the authentication API responds with anything else, a message box is presented denying access.

10

15

Fig. 9 illustrates one embodiment of the present invention for one or more different types of transactions. As shown in this example, a centralized authentication system 81 is used to authenticate users for a variety of different transactions, including both business and consumer mass markets. Four examples of such transactions include point of sale transactions 82 (e.g. credit card purchases at retail stores), system and/or information access for businesses 83 (e.g. authentication for a company's network), financial institutions providing consumer services 84 (e.g. on-line banking, account transfers, payment, investments, ATM access, traditional banking services, and the like), and electronic commerce 85 (e.g. Internet sales). Beyond these four examples, other types of transactions are also contemplated.

20

25

One or more agent machines are clients to the central authentication system 81. Typically, an agent machine is dedicated to only one type of transaction, however, it should be realized that any given agent machine may be associated with a plurality of different types of transactions. Because each of the various types of transactions authenticate using the central authentication system 81, a user has a single and uniform authentication methodology for each of the various types of transactions. Beyond the convenience of a uniform authentication methodology, improved authentication can also be realized. Currently, many consumer related transactions require only a PIN or password (e.g. ATM, on-line banking, etc.). In some cases, such as retail credit card transactions, the only authentication sometimes involves a clerk comparing a signature, if this duty is performed at all. If a strong authentication system is used, such a two-factor

30

35

systems using a PIN in combination with a token or biometric reading, authentication for many systems will be dramatically improved.

A variety of business models can be developed around the present technology. For instance, fees can be collected, either from the user or the business, for each authentication transaction performed by the central authentication system. Fee structures can vary widely, including monthly or annual rates for access to the central authentication service, flat rates for each transaction, variable rates based on the value of the transaction, variable rates based on authentication volume, any combination of the foregoing, and the like. Further, revenue generation through sales and/or leases of token generating devices could complement or replace authentication fees.

EXAMPLES

The following examples depict three typical scenarios of users interacting with the authentication system of the present invention. Example 1 depicts an exemplary interaction in which the authentication server is in "Normal" mode. Example 2 depicts an exemplary illustration in which the authentication server is in "New PIN" mode. Finally, Example 3 depicts an exemplary interaction in which the authentication server is in "Next Token" mode.

Example 1

As depicted in Fig. 10, the authentication client 21 begins the interaction by requesting a Login ID, Password and SecurID Number. After the user has entered the information and clicks on "OK", the authentication client concatenates the Password and SecurID Number as the Passcode. The authentication client then sends the Login ID and Passcode to the authentication server. If the correct information was entered and the Passcode is valid, the authentication server returns the error code "ACM_OK". The authentication client receives the error code and presents the user with a dialog box informing the user of the successful authentication interaction as depicted in Fig. 11.

Example 2

In this example, an administrative user has configured the authentication server to be in a state of "New PIN" mode. As depicted in Fig. 12, the authentication client 21 begins the interaction by requesting a Login ID, Password (PIN) and SecurID Number. After the user has entered the information and clicks on "OK", the authentication client concatenates the Password and SecurID Number as the Passcode. The authentication

client then sends the Login ID and Passcode to the authentication server. Since the user's account is in "New PIN" mode, the authentication server returns the error code "ACM_ACCESS_DENIED". The authentication client then retries the authentication request, but this time only sends a Passcode containing the SecurID Number (i.e., no PIN number). If the revised Passcode (i.e. SecurID) correctly corresponds to the Login ID, the authentication server returns the error code "ACM_NEW_PIN_REQUIRED". The authentication client then displays a dialog box to the user as depicted in Fig. 13, requesting the user to reconfirm the Password (PIN). If the user enters a Password that matches the password in the original dialog of Fig. 12, then the PIN is sent to the authentication server and is set for the user's account. The authentication server, then changes the state from "New PIN" mode to "Normal" mode. If the user does not enter a password that matches the previously entered password, the authentication client displays, as depicted in Fig. 14, a dialog box informing the user of the failure to match. Once the user has acknowledged the dialog box, the authentication client prompts the user to enter a new Password and reconfirm the password as depicted in Fig. 15. The authentication client compares the new PIN with the reconfirmation PIN, and sends the new PIN to the authentication server when the PIN numbers match. Once the authentication server is sent the new PIN, the authentication server returns an error code of "ACM_SUCCESS".

Example 3

In this example, the authentication server is in "Next Token" mode. As depicted in Fig. 16, the authentication client 21 begins the interaction by requesting a Login ID, Password and SecurID (Token) Number. After the user has enter the information and clicks on "OK", the authentication client concatenates the Password and SecurID Number as the Passcode. The authentication client then sends the Login ID and Passcode to the authentication server. Since the user's account is in "Next Token" mode, the authentication server will return the error code "ACM_NEXT_CODE_REQUIRED" to the client. The authentication client, as depicted in Fig. 17, prompts the user with the previous SecurID number (token) and asks the user to enter the next SecurID Number once it changes. Once the user enters the next SecurID Number and clicks "OK", the SecurID Number is sent to the authentication server. The authentication server determines whether the SecurID Number is correct, and if so returns an error code to the authentication client of "ACM_SUCCESS". The authentication client receives the error

code and presents the user with a dialog box informing the user of the successful authentication interaction as depicted in Fig. 18.

5 The foregoing description of the preferred embodiment of the invention and examples have been presented for purposes of illustration and description. It is not intended to be exhaustive nor to limit the invention to the precise form disclosed. Many alternatives, modifications, and variations will be apparent to those skilled in the art in light of the above teaching. Accordingly, this invention is intended to embrace all alternatives, modifications, and variations that fall within the spirit and broad scope of the amended claims.

WHAT IS CLAIMED IS:

1. A computer system, characterized in that it comprises:
 - a) a network of computers;
 - b) an authentication server on the network, said authentication server comprising
 - i) an authentication engine having a first state wherein the authentication engine uses a password comprising two elements and a second state wherein the authentication engine uses a password comprising one element; and
 - ii) an application programming interface through which other computers of the network may make requests to the authentication engine;
 - c) a plurality of agent computers on the network, each of said plurality of agent computers comprising agent software for the authentication server being encoded for
 - i) sending a first authentication request via the application programming interface to the authentication engine using a password comprising two elements, and
 - ii) if authentication is not established in response to the first authentication request, sending a second authentication request via the application programming interface to the authentication engine using a password comprising one element.
2. The computer system of claim 1, wherein the two elements of the password in the first state are a user identifier and a token.
3. The computer system of claim 2, wherein the password in the first state is a string comprising the concatenation of the user identifier and the token.
4. The computer system of claim 2, wherein a token generation device and authentication engine independently generate the token in parallel.
5. The computer system of claim 2, wherein the user identifier is a PIN.
6. The computer system of claim 5, wherein the second state is new PIN mode.

7. The computer system of claim 2, wherein the one element of the password for the second state is a token.
8. The computer system of any of the preceding claims, wherein the authentication engine is a centrally managed, strong, two-factor user authentication service.
9. The computer system of any of the preceding claims, wherein one or more of the plurality of agent computers are servers on the network.
10. A method for centralized user authentication on a computer network where each of a plurality of agent computers perform the method, characterized in that the method comprises the steps of:
 - a) providing an authentication agent on each of the agent computers;
 - b) prompting a user through the authentication agent to enter a user identification and token;
 - c) receiving the user identifier and token from the user;
 - d) concatenating the user identifier and token;
 - e) passing the concatenated user identifier and token to a central authentication system on the network;
 - f) receiving a response from the central authentication system; and
 - g) if the response does not permit access, passing the token as a password to the central authentication system.
11. The method of claim 10, wherein in step (e) the concatenated user identifier and token are passed as a password to the central authentication system.
12. The method of claim 10 or 11, wherein in step (g) the token is passed as a password if the response denies access.
13. The method of claim 10, 11, or 12, wherein in step (c), a login ID is additionally obtained.
14. The method of claim 10, 11, 12, or 13, wherein in step (e), the login ID is additionally passed to the central authentication system.
15. A computer readable medium, comprising instructions for performing the method for centralized user authentication on a computer network where each of a plurality of agent

computers perform the method, characterized in that the instructions comprise the steps of:

- a) providing an authentication agent on each of the agent computers;
- b) prompting a user through the authentication agent to enter a user identification and token;
- c) receiving the user identifier and token from the user;
- d) concatenating the user identifier and token;
- e) passing the concatenated user identifier and token to a central authentication system on the network;
- f) receiving a response from the central authentication system; and
- g) if the response does not permit access, passing the token as a password to the central authentication system.

16. A program product, comprising signal bearing media embodying a program of machine readable instructions executable by a digital data processor to perform a method for centralized user authentication on a computer network where each of a plurality of agent computers perform the method, characterized in that the instructions comprise the steps of:

- a) providing an authentication agent on each of the agent computers;
- b) prompting a user through the authentication agent to enter a user identification and token;
- c) receiving the user identifier and token from the user;
- d) concatenating the user identifier and token;
- e) passing the concatenated user identifier and token to a central authentication system on the network;
- f) receiving a response from the central authentication system; and
- g) if the response does not permit access, passing the token as a password to the central authentication system.

17. A method for centralized user authentication on a computer network, characterized in that the method comprises the steps of:

- a) providing a central authentication system, wherein the central authentication system comprises an authentication engine;
- b) receiving at the authentication engine a user identifier and a token corresponding to a user from an agent computer of the central authentication system;

c) if the central authentication system requires the user identifier and token, processing by the authentication engine the user identifier and token to determine whether the user is authenticated; and

d) if the central authentication system requires only the token, processing by the authentication engine the token to determine whether the user is authenticated.

18. The method of claim 17, wherein the authentication engine extracts the token from the concatenated value to determine the value of the token through a reverse hashing algorithm utilizing the user identifier.

1 / 10

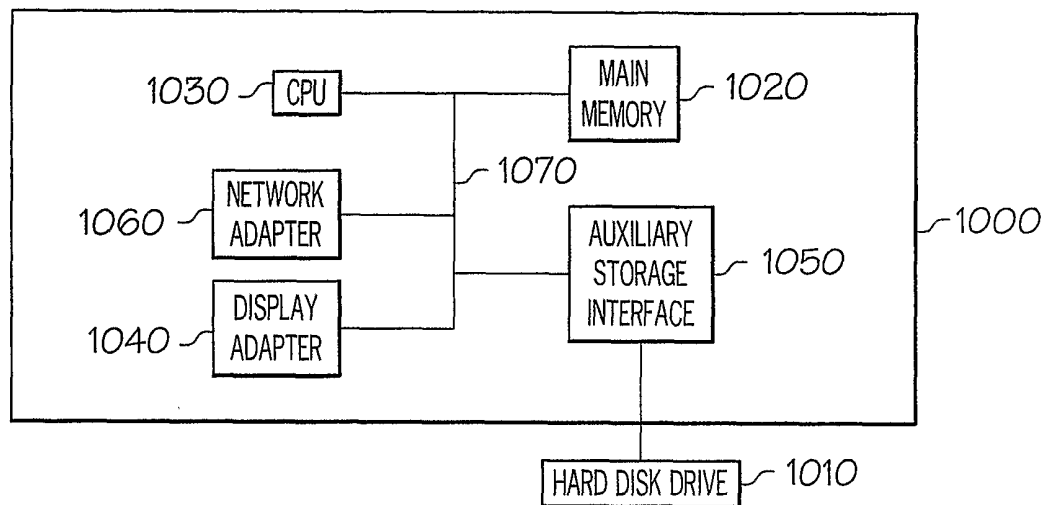


FIG. 1

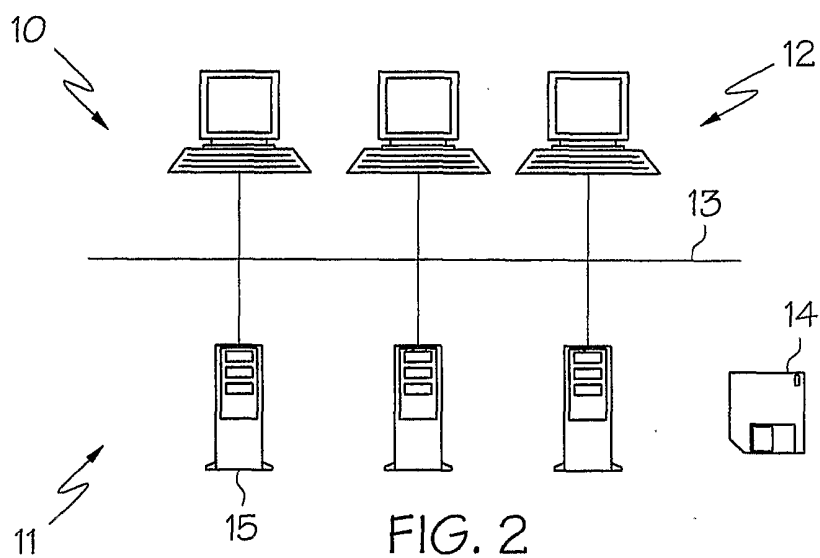


FIG. 2

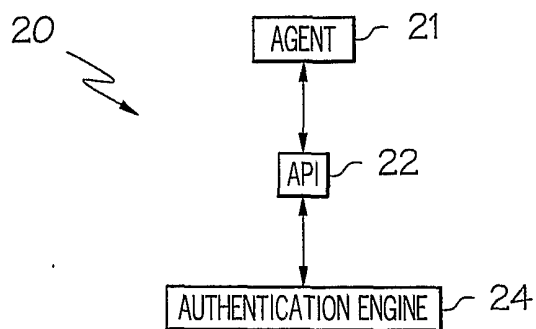


FIG. 3

2 / 10

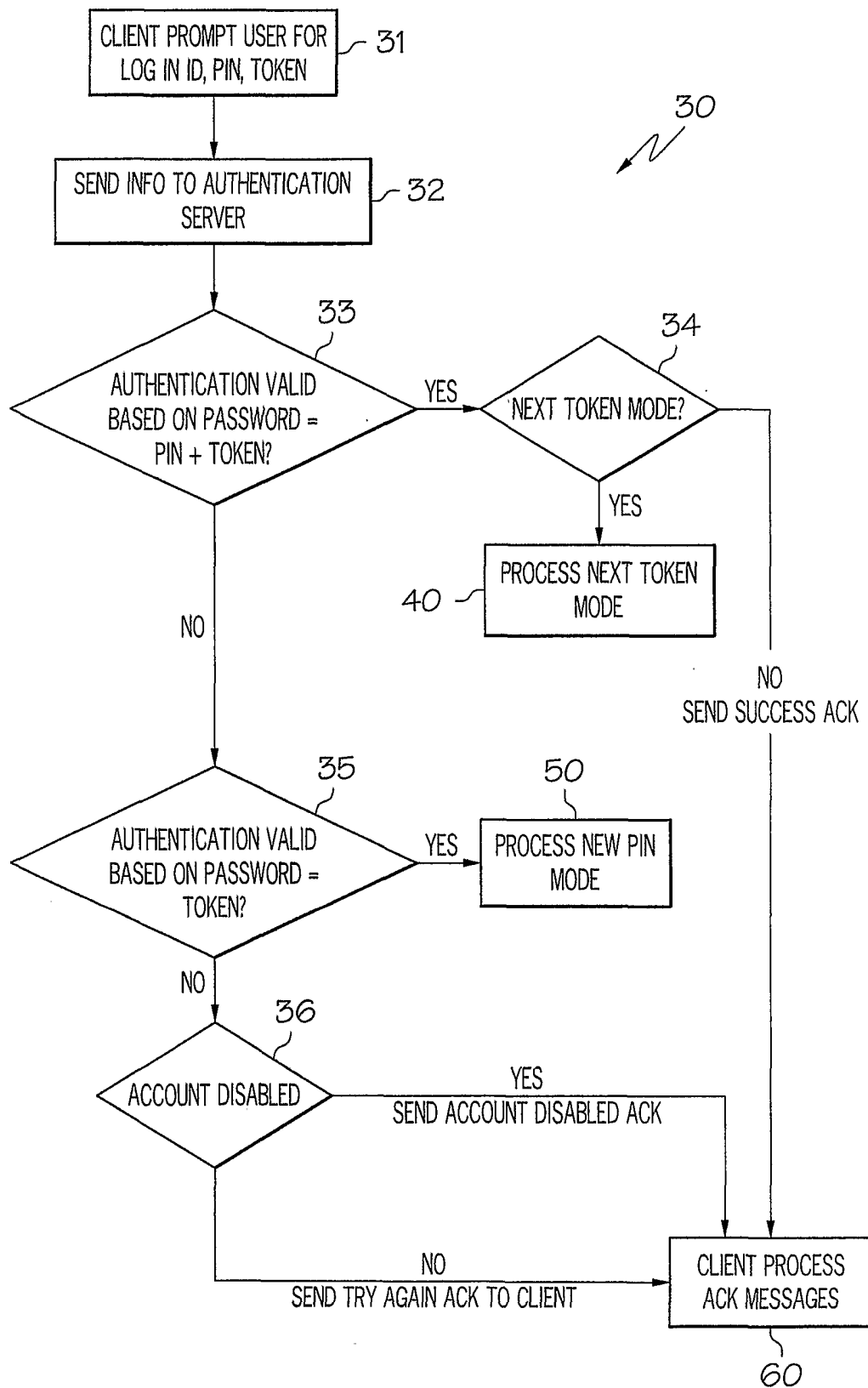


FIG. 4

3 / 10

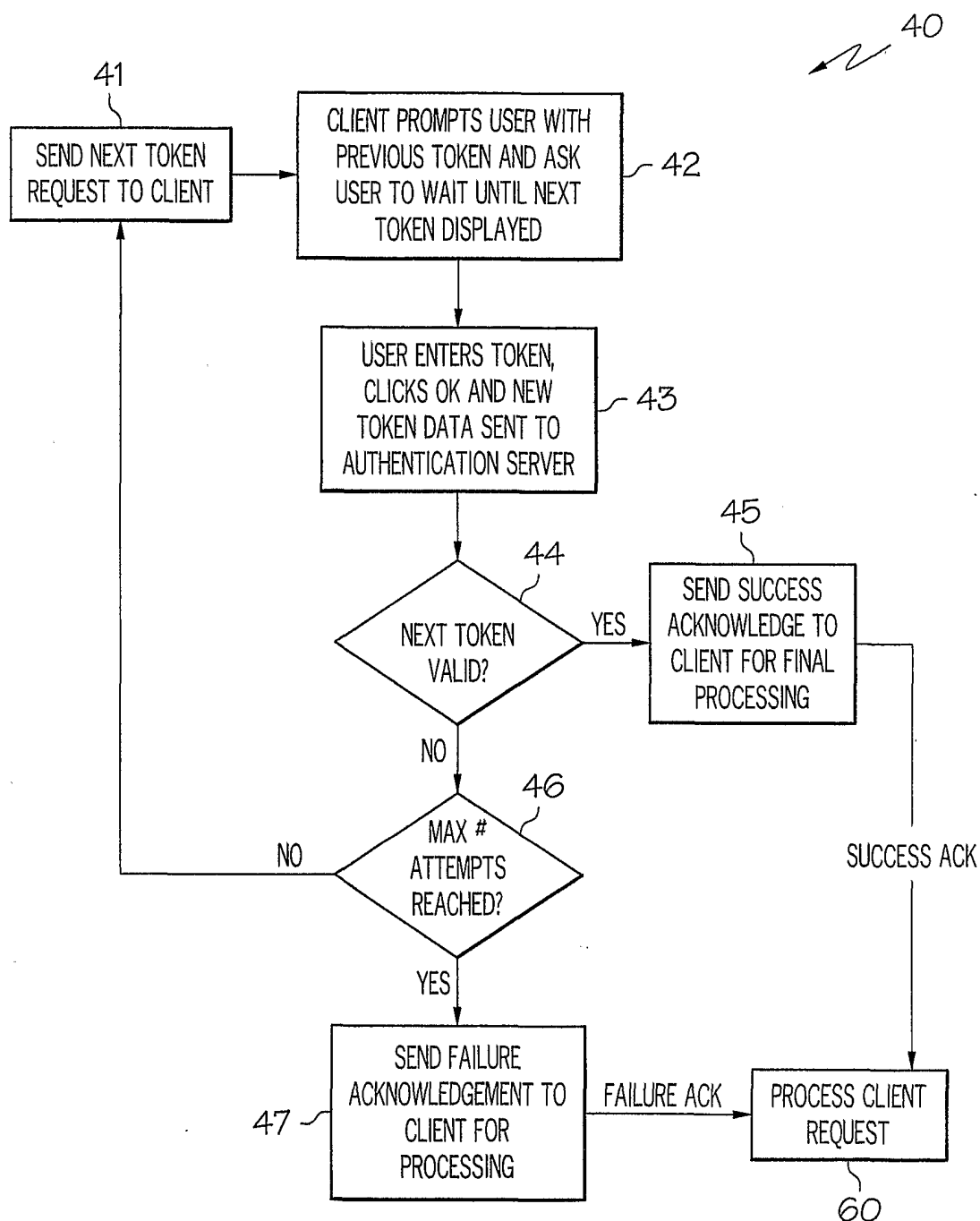


FIG. 5

4 / 10

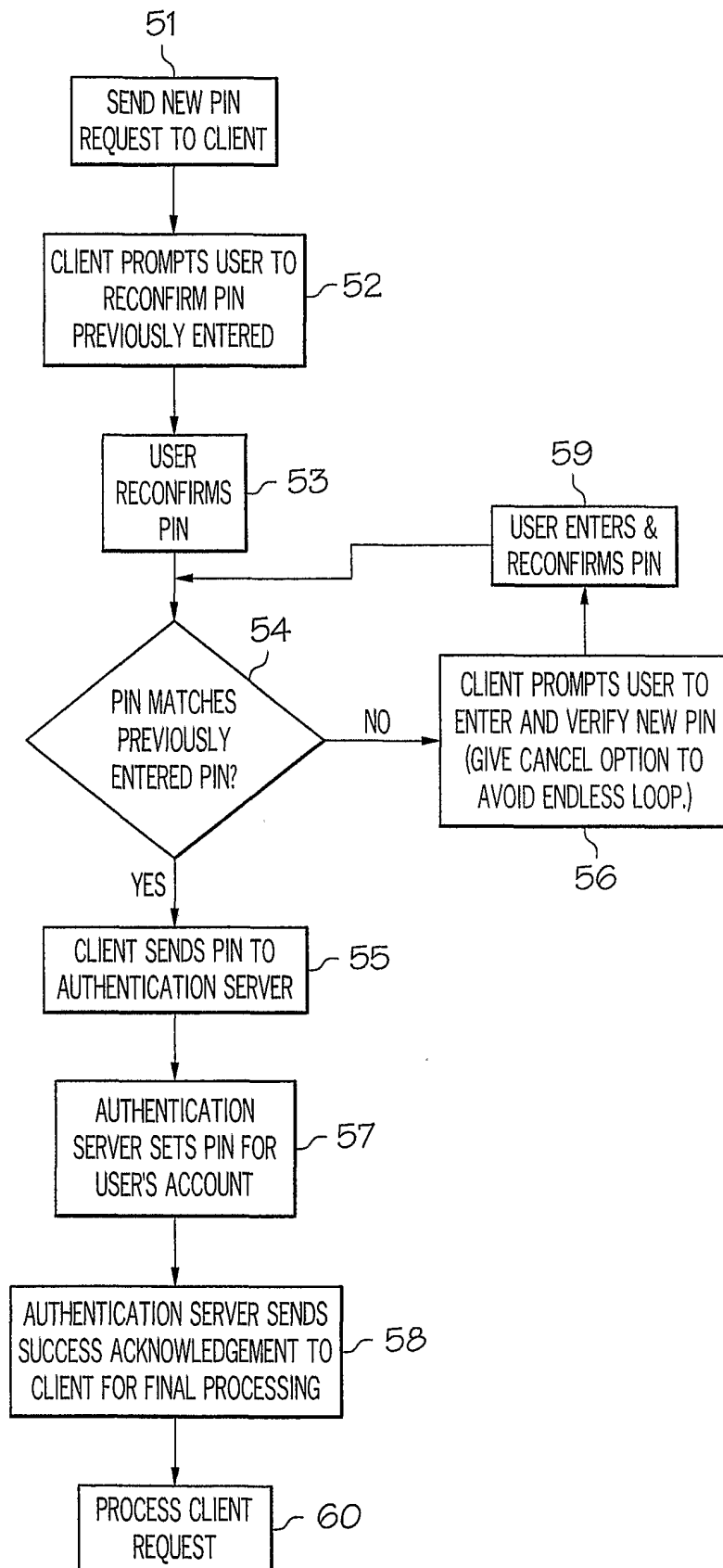


FIG. 6

5 / 10

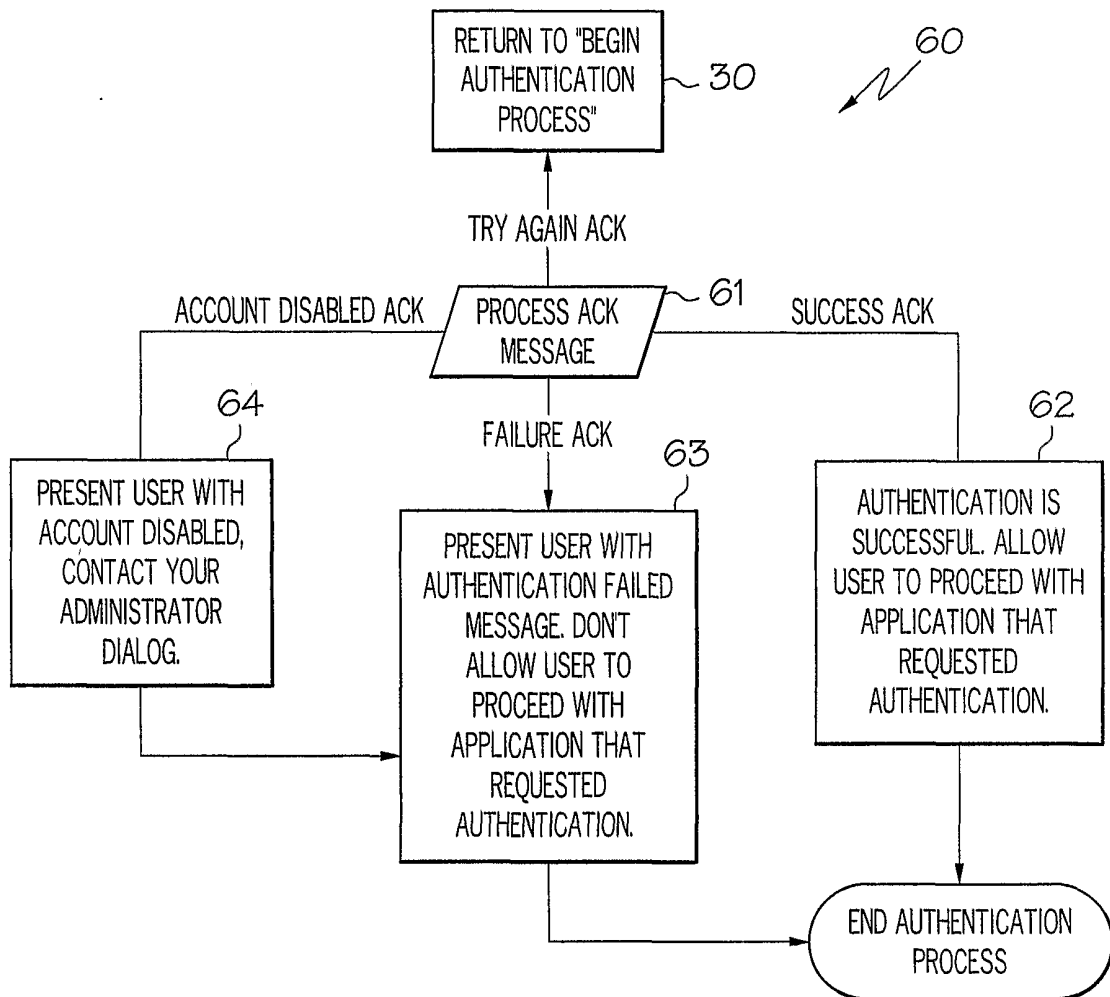


FIG. 7

6 / 10

The figure shows a graphical user interface window titled "SMA CLIENT". The window is divided into several sections. On the left, there is a large rectangular area labeled "<INSERT ANIMATED GRAPHIC HERE>". To the right of this, at the top, is another rectangular area labeled "<INSERT CONTEXT SENSITIVE HELP HERE>". Below these, on the right side, is a section titled "LOGIN INFORMATION" which contains three input fields: "LOGIN ID / USER ID:", "PIN / PASSWORD:", and "TOKEN:". At the bottom of the window, there are three buttons: "HELP" on the left, and "OK" and "CANCEL" on the right.

FIG. 8

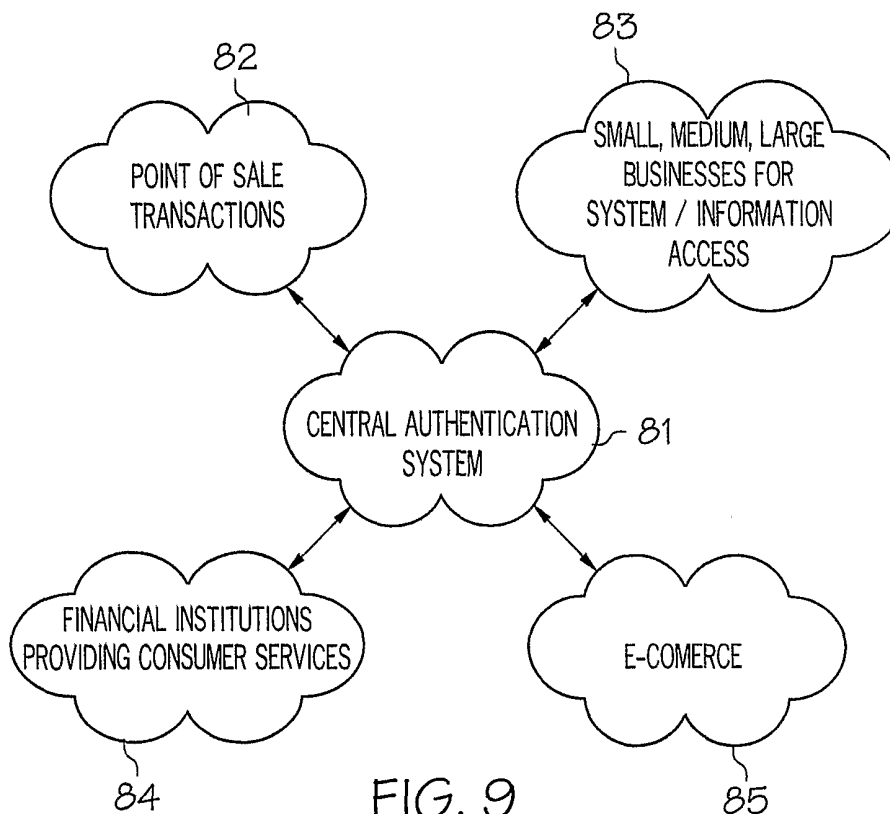
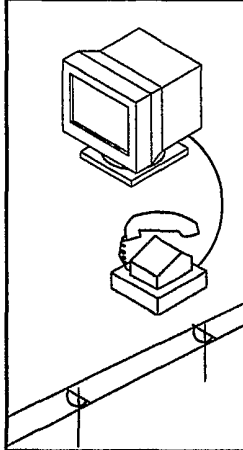


FIG. 9

7 / 10

PROCTOR & GAMBLE SMA AUTHENTICATION



ENTER YOUR SECURID
CARD NUMBER:

00000000

SECURID

THE NUMBER ON THE CARD
CHANGES EVERY 60 SECONDS. IF THE NUMBER CHANGES
WHILE YOU ARE TYPING IT IN, SIMPLY DELETE THE CURRENT
NUMBER AND ENTER THE NEW NUMBER

ENTER LOGIN INFORMATION

LOGIN ID:

PASSWORD:

SECURID NUMBER:

FIG. 10

SMACONNECT SUCCESSFUL

THE USER IS AUTHENTICATED SUCCESSFULLY.

FIG. 11

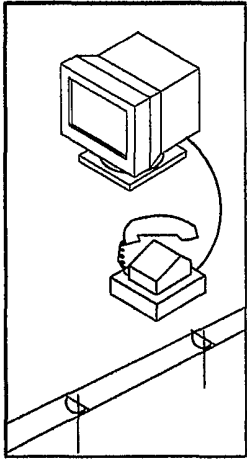
SMACONNECT SUCCESSFUL

THE USER IS AUTHENTICATED SUCCESSFULLY.

FIG. 18

8 / 10

PROCTOR & GAMBLE SMA AUTHENTICATION



ENTER YOUR SECURID
CARD NUMBER:

1888888888

SECURID

THE NUMBER ON THE CARD
CHANGES EVERY 60 SECONDS. IF THE NUMBER CHANGES
WHILE YOU ARE TYPING IT IN, SIMPLY DELETE THE CURRENT
NUMBER AND ENTER THE NEW NUMBER

ENTER LOGIN INFORMATION

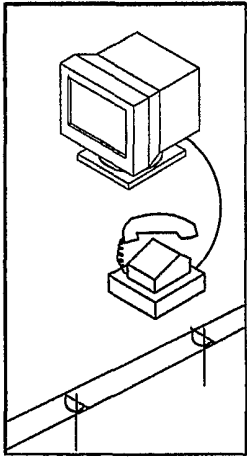
LOGIN ID:

PASSWORD:

SECURID NUMBER:

FIG. 12

PROCTOR & GAMBLE SMA AUTHENTICATION



SET YOUR PERSONAL IDENTIFICATION NUMBER (PIN):

YOUR PIN IS A SECRET PASSWORD THAT ONLY YOU KNOW. IT
IS SIMILAR TO THE PIN YOU USE WITH YOUR BANK CARD. THE
PIN MUST BE AT LEAST 4 CHARACTERS LONG, BUT NO MORE THAN
8 CHARACTERS LONG.

RE-ENTER PASSWORD

PASSWORD:

FIG. 13

9 / 10

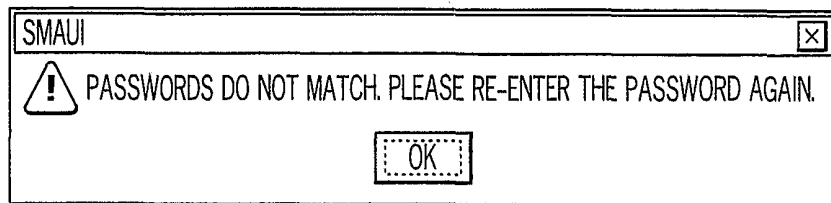


FIG. 14

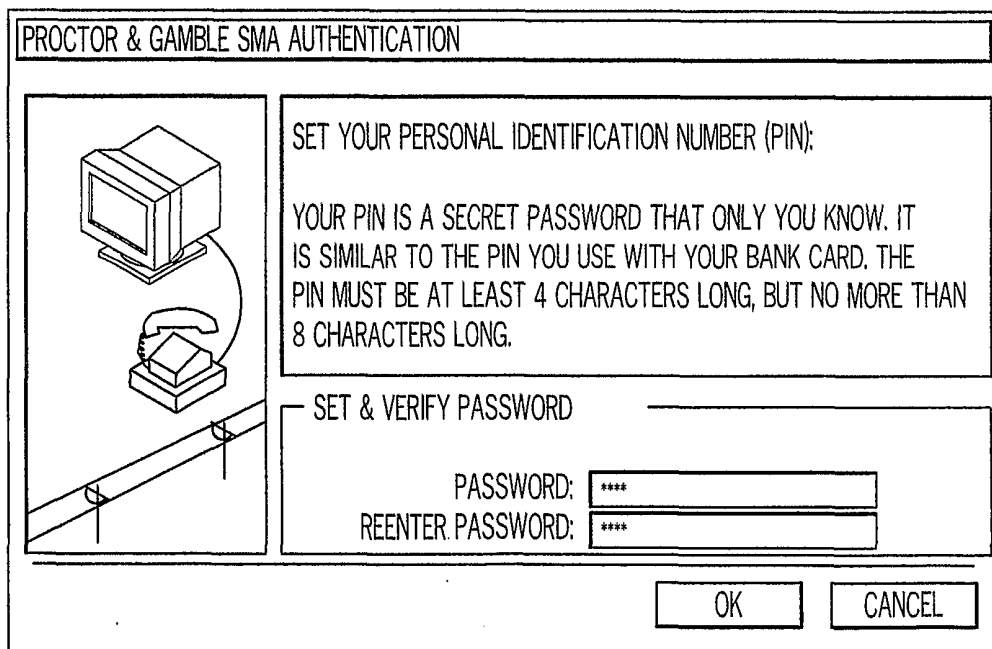
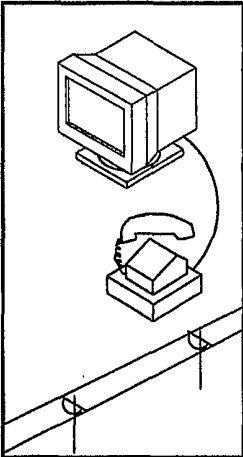


FIG. 15

10 / 10

PROCTOR & GAMBLE SMA AUTHENTICATION



ENTER YOUR SECURID
CARD NUMBER:

THE NUMBER ON THE CARD
CHANGES EVERY 60 SECONDS. IF THE NUMBER CHANGES
WHILE YOU ARE TYPING IT IN, SIMPLY DELETE THE CURRENT
NUMBER AND ENTER THE NEW NUMBER

ENTER LOGIN INFORMATION

LOGIN ID: tm2496

PASSWORD: ****

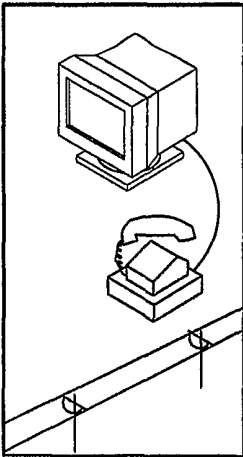
SECURID NUMBER: 758660

OK

CANCEL

FIG. 16

PROCTOR & GAMBLE SMA AUTHENTICATION



ENTER YOUR SECURID
CARD NUMBER:

THE NUMBER ON THE CARD
CHANGES EVERY 60 SECONDS. IF THE NUMBER CHANGES
WHILE YOU ARE TYPING IT IN, SIMPLY DELETE THE CURRENT
NUMBER AND ENTER THE NEW NUMBER

ENTER NEXT SECURID NUMBER

PREVIOUS SECURID NUMBER: 758660

NEXT SECURID NUMBER: 021672

OK

CANCEL

FIG. 17