(12) PATENT (11) Application No. AU 199655615 B2
(19) AUSTRALIAN PATENT OFFICE (10) Patent No. 720200

(54) Title
Electronic-monetary system

(51)[7] International Patent Classification(s)
G07F 007/08 **G06F 017/60**

(21) Application No: **199655615** (22) Application Date: **1996.04.19**

(87) WIPO No: **WO96/33476**

(30) Priority Data

(31) Number	(32) Date	(33) Country
08/427287	**1995.04.21**	**US**

(43) Publication Date : **1996.11.07**
(43) Publication Journal Date : **1997.01.02**
(44) Accepted Journal Date : **2000.05.25**

(71) Applicant(s)
Citibank, N.A.

(72) Inventor(s)
Sholom S. Rosen

(74) Agent/Attorney
BALDWIN SHELSTON WATERS,Level 21,60 Margaret Street,SYDNEY NSW 2000

(56) Related Art
US 557518
WO 93/10503
WO 93/08545

OPI DATE 07/11/96 APPLN. ID 55615/96
AOJP DATE 02/01/97 PCT NUMBER PCT/US96/05521

AU9655615

(51) International Patent Classification 6 : G07F 7/08, G06F 17/60

A2

(11) International Publication Number: WO 96/33476

(43) International Publication Date: 24 October 1996 (24.10.96)

(21) International Application Number: PCT/US96/05521

(22) International Filing Date: 19 April 1996 (19.04.96)

(30) Priority Data:
08/427,287 21 April 1995 (21.04.95) US

(71) Applicant: CITIBANK, N.A. [US/US]; 399 Park Avenue, New York, NY 10043 (US).

(72) Inventor: ROSEN, Sholom, S.; Apartment 7A, 10 West 86th Street, New York, NY 10024 (US).

(74) Agents: DOWLING, Thomas, P. et al.; Morgan & Finnegan, L.L.P., 345 Park Avenue, New York, NY 10154 (US).

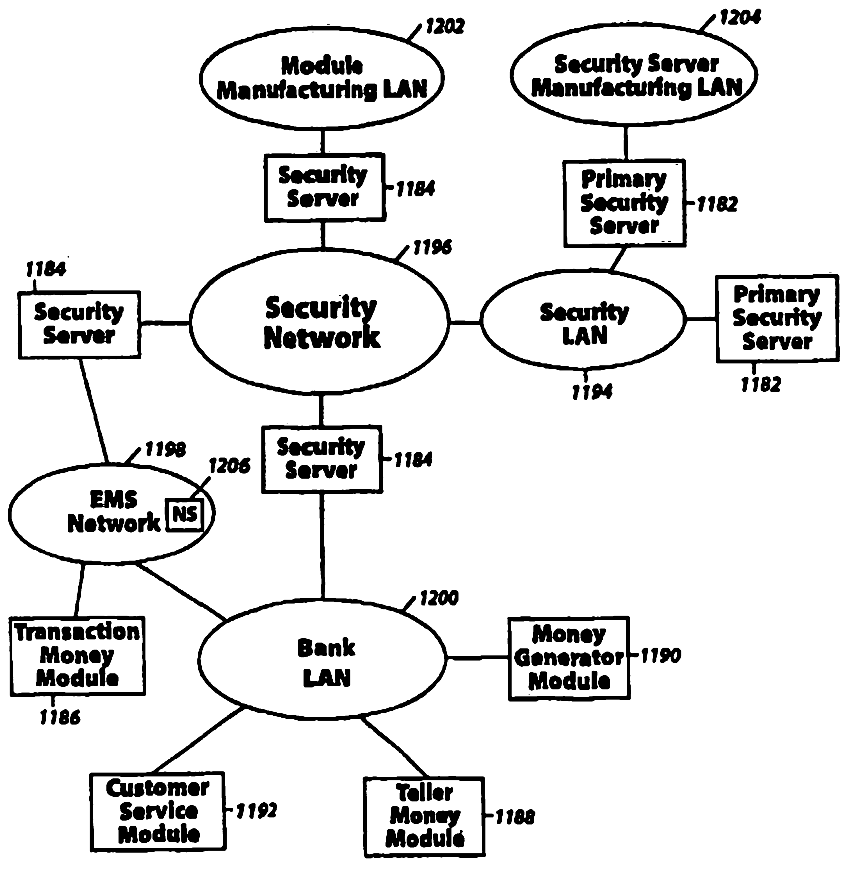
(81) Designated States: AL, AM, AT, AU, AZ, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian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

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: ELECTRONIC-MONETARY SYSTEM

(57) Abstract

An electronic-monetary system having (1) banks or financial institutions that are coupled to a money generator device for generating and issuing to subscribing customers electronic money including electronic currency backed by demand deposits and electronic credit authorizations; (2) correspondent banks that accept and distribute the electronic money; (3) a plurality of transaction devices that are used by subscribers for storing electronic money, for performing money transactions with the on-line systems of the participating banks or for exchanging electronic money with other like transaction devices in off-line transactions; (4) teller devices, asscoiated with the issuing and correspondent banks, for process handling and interfacing the transaction devices to the issuing and correspondent banks, and for interfacing between the issuing and correspondent banks themselves; (5) a clearing bank for balancing the electronic money accounts of the different issuing banks; (6) a data communications network for providing communications services to all components of the system; and (7) a security arrangement for maintaining the integrity of the system, and for detecting counterfeiting and tampering within the system. An embodiment of the invention includes a customer service module which handles lost money claims and links accounts to money modules for providing bank access.



FIELD OF THE INVENTION

The present invention relates to an electronic monetary system for implementing electronic money transactions as an alternative medium of economic exchange to cash, checks, credit and debit cards, and electronic funds transfer.

BACKGROUND OF THE INVENTION

Today, approximately 350 billion coin and currency transactions occur between individuals and institutions every year. The extensive use of coin and currency transactions has limited the automation of individual transactions such as purchases, fares, and bank account deposits and withdrawals. Individual cash transactions are burdened by the need of having the correct amount or providing change therefor. Furthermore, the handling, and managing (including guarding) of paper cash and coins is inconvenient, costly and time consuming for both individuals and financial institution alike. It is estimated that in the United States alone, $60 billion dollars a year is spent by money handlers simply to move money. In addition, the security of paper money is seriously threatened by the relative ease of counterfeiting using, for example, widely accessible, high quality color copiers.

Although checks may be written for any specific amount up to the amount available in the account, checks have very limited transferability and must be supplied from a physical inventory. Paper-based checking systems do not offer sufficient relief from the limitations of cash transactions, sharing many of the inconvenience of handling currency while adding the inherent delays associated with processing checks. To this end, economic exchange has striven for greater convenience at a lower cost, while also seeking improved security.

Automation has achieved some of these qualities for large transactions through computerized electronic funds transfer ("EFT") systems. Electronic funds transfer is essentially a process of value exchange achieved through the banking system's centralized computer transactions. EFT services are a transfer of payments utilizing electronic "checks," which are used primarily by large commercial organizations.

The Automated Clearing House (ACH) and point of sale (POS) systems are examples of electronic funds transfer systems that have become used by retail and commercial organizations on a substantial basis in recent years. However, the payments made through these types of EFT systems are limited in that they cannot be performed without the banking system. Moreover, ACH transactions usually cannot be performed during off business hours.

Home Banking bill payment services are examples of an electronic funds transfer system used by individuals to make payments. Currently, home banking initiatives have found few customers. Of the banks that have offered services for payments, account transfers and information over the telephone lines using personal computers, less than one percent of the bank's customers are using the service. One reason that Home Banking has not been a successful product is because the customer cannot deposit and withdraw money as needed in this type of system.

Current EFT systems, credit cards, or debit cards, which are used with an on-line system to transfer money between accounts, such as between the account of a merchant and that of a customer, cannot satisfy the need for an automated transaction system that provides for the transfer of universally accepted economic value outside of the banking system. Moreover, credit and debit card systems are generally insecure against fraud and do not provide for adequate privacy.

To implement an automated, yet more convenient transaction system that does not require the banking system to intermediate the transfer, and that can dispense some form of economic value, there has been a trend towards off-line electronic funds transfer. For example, numerous ideas have been proposed for some form of "electronic money" that can be used in cashless payment transaction as alternatives to the traditional currency and check types of payment systems. See U.S. Patent No. 4,977,595, entitled "METHOD AND APPARATUS FOR IMPLEMENTING ELECTRONIC CASH," and U.S. Patent No. 4,305,059, entitled "MODULAR FUNDS TRANSFER SYSTEM."

The more well known techniques include magnetic stripe cards purchased for a given amount and from which a prepaid value can be deducted for specific purposes. Upon exhaustion of the economic value, the cards are thrown away. Other examples include memory cards or so called smart cards which are capable of repetitively storing information representing value that is likewise deducted for specific purposes.

However, these proposed systems suffer from a failure to recognize fully the significance of bank deposits as money, and their necessity to back any form of universally accepted monetary representations that may be issued. In these systems representations of economic value, whether electronic or paper, are issued without the backing of equal valued liabilities as the counterpart to their assets.

None of these proposed paperless payment systems are comprehensive enough so as to implement a multipurpose electronic monetary system that includes not only the automated devices that allow subscribers to transfer electronic funds or money between them without any intermediating system, but that also encompasses and includes an entire banking system for generating the value represented by the electronic money and for clearing and settling the electronic money accounts of the banks and financial institutions involved to maintain a monetary balance within the system.

Thus, there is a need for a system that allows common payor to payee economic exchanges without the intermediation of the banking system, and that gives control of the payment process to the individual. Furthermore, a need exists for providing a system of economic exchange that can be used by large organizations for commercial payments of any size, that does not have the limitations of the current EFT systems.

SUMMARY OF THE INVENTION

It is an object of the present invention to overcome or ameliorate one or more of the disadvantages of the prior art, or at least to provide a useful alternative.

According to the invention, there is provided a method for a first subscriber to exchange an electronic representation of a first foreign currency stored in a first transaction module for an electronic representation of a second foreign currency stored in a second transaction module, comprising the steps of:

(a) establishing a cryptographically secure session between said first transaction module and said second transaction module;

(b) said subscriber selecting, by way of said first transaction module, a first amount of said first foreign currency to be sold;

(c) checking if said first transaction module has sufficient funds;

(d) said first transaction module sending said first amount to said second transaction module, via said cryptographically secure session;

(e) said second transaction module prompting its owner to select an exchange rate or a second amount of said second currency;

(f) checking if said second transaction module has sufficient funds;

(g) said second transaction module sending a message indicative of said second amount or said exchange rate to said first transaction module, via said cryptographically secure session;

(h) said first subscriber confirming said second amount or said exchange rate;

(i) said first transaction module sending said electronic representation of first foreign currency to said second transaction module, in said first amount, via said cryptographically secure session;

(j) said second transaction module sending said electronic representation of said second foreign currency to said first transaction module, in said second amount, via said cryptographically secure session; and

(k) committing said first transaction module to transferring said first foreign currency to said second transaction module and said second transaction module to receiving said first foreign currency from said first transaction module, and committing said second transaction module to transferring said second foreign currency to said first transaction module and said first transaction module to receiving said second foreign currency from said second transaction module, in an order which is not predictable.

A method and apparatus in accordance with a preferred embodiment of the present invention employ an electronic-monetary system having (1) banks or financial institutions that are coupled to a money generator device for generating and issuing to subscribing customers electronic money including electronic currency backed by demand deposits and electronic credit authorizations; (2) correspondent banks that accept and distribute the electronic money; (3) a plurality of transaction devices that are used by subscribers for storing electronic money, for performing money transactions with the on-line systems of the participating banks or for exchanging electronic money with other like transaction devices in off-line transactions; (4) teller devices, associated with the issuing and correspondent banks, for process handling and interfacing the transaction devices to the issuing and correspondent banks, and for interfacing between the issuing and correspondent banks themselves; (5) a clearing bank for balancing the electronic money accounts of the different issuing banks; (6) a data communications network for providing communications services to all components of the system; and (7) a security arrangement for maintaining the integrity of the system, and for detecting counterfeiting and tampering within the system.

In the preferred embodiment, the functions of the money generating devices, the transaction devices, and the teller devices will be performed by a combination of tamper-proof computer hardware and application software modules that may be networked together. Information is transmitted in an encrypted form to provide security from unauthorized inspection. The electronic money is transmitted with digital signatures to provide authentication, and security from modification or counterfeiting.

The electronic money exchanged by these devices may be an electronic representation of currency or credit. In one aspect, the electronic currency is equivalent of bank notes and is interchangeable with conventional paper money through claims on deposits in an issuing bank, but can be withdrawn or deposited both at an issuing bank and at a correspondent bank. However, only the issuing banks can generate the electronic currency, and will be liable for its redemption.

The issuing banks later utilize inter-bank clearing and settling processes to maintain the monetary balance in the banking system, as is currently practiced by today's banking industry.

The electronic money representations are fungible, universally accepted, and undeniably redeemable from the issuing banks, i.e., they have the characteristics of money transactions. To preserve the integrity of the electronic monetary system, each exchange of electronic money includes, along with other information, data identifying the monetary unit of the credit or currency, (i.e., dollars, yen, etc.) the amount by unit of credit or currency, the bank issuing the electronic credit or currency, and several digital signatures.

According to preferred embodiments of the invention, an electronic monetary system provides for transactions utilizing electronic money including electronic currency backed by demand deposits in a bank in lieu of cash transactions, and electronic credit authorizations. In a preferred embodiment of the present invention, the EMS comprises a money module for generating the electronic money; a money module for issuing, distributing, and accepting the electronic money; and a money module for accepting, storing, and transferring the electronic money between other accepting money modules and between the accepting money module and the issuing money module.

According to further preferred embodiments of the invention, an electronic monetary system is provided for implementing and maintaining electronic money which includes electronic currency that is interchangeable with conventional money through claims on deposits in a bank and electronic credit authorizations.

In a preferred embodiment of the present invention, the system includes a plurality of issuing banks; a generator module for creating electronic money; teller modules coupled to the generator modules, for performing teller transactions and for interacting with other teller modules, such transactions including the accepting and the distributing of the electronic money; a security system for providing the overall integrity of the electronic monetary system; a clearing and settling process for balancing the electronic money accounts of the separate issuing banks and for clearing the electronic money issued by the issuing banks; and a plurality of transaction modules owned by authorized users, for transferring the electronic money between the transaction modules and between the transaction modules and the teller modules; and a customer service module handling lost money claims and for linking accounts to money modules to provide bank access.

Preferably, the functions of the generator modules, the transaction modules, the teller modules, and the customer service modules will be performed by a combination of tamper-proof computer hardware and application software that may be networked together. The electronic money exchanged by these modules, which may be an electronic representation of currency backed by demand deposit accounts at the issuing bank or credit authorizations, may be transmitted with digital signatures to provide security from unauthorized modification or counterfeiting. In a preferred embodiment, security from counterfeiting and tampering is also provided by requiring the modules and the individual units of electronic money to be renewed periodically. Preferably, transfer of a note between modules includes a sequence identification number to facilitate note transfer history reconciliation. Offending modules or counterfeit electronic money can be removed from circulation as soon as they are discovered.

Also, preferred embodiments of the invention provide a process to claim lost money whereby a transaction module owner/holder may submit claims to the banking system via a customer service module, and subsequent to note transfer history reconciliation by the banking system, the owner/holder may recover any lost notes that were claimed.

Preferred embodiments of the invention provide a process for a transaction module to update credit such that there is only one credit note for each credit account in a money module.

Preferably also, a process is provided for an owner/holder of a transaction module to securely and reliably purchase goods from a merchant.

Unless the context clearly requires otherwise, throughout the description and the claims, the words 'comprise', 'comprising', and the like are to be construed in an inclusive as opposed to an exclusive or exhaustive sense; that is to say, in the sense of "including, but not limited to".

BRIEF DESCRIPTION OF THE DRAWINGS

Additional aspects, features, and advantages of the invention will be understood and will become more readily apparent when the invention is considered in the light of the following description made in conjunction with the accompanying drawings, wherein:

Figure 1A is a diagram showing the EMS Security Hierarchy;

Figure 1B is a diagram showing the security network messaging between a primary security server and an ordinary security server;

Figure 2 is a diagram showing the security network structure for the EMS;

Figure 3A illustrates the functional components of a security server;

Figure 3B illustrates the functional components of a network server;

Figure 4A illustrates the functional components of a customer service module;

Figure 4B illustrates the functional components of a primary security server;

Figure 5 shows an overview of the network sign-on procedure;

Figures 6A-6K illustrate a Network Sign-On protocol;

Figures 7A-7E illustrate an Establish Session protocol in the EMS;

Figures 8A-8B illustrate a Transfer Notes protocol;

Figures 9A-9D illustrate a Foreign Exchange protocol;

Figure 10 illustrates a Commit protocol for modules in the EMS;

Figures 11A-11B illustrate an Abort Transaction protocol for modules in the EMS;

Figures 12A-12C illustrate a Point of Sale (POS) Payment protocol;

Figures 13A-13B illustrate an Update Credit form Issuing Bank protocol;

Figure 14 illustrate a setup Credit protocol;

Figures 15A-15B illustrates a Request Credit protocol;

Figure 16 shows an example of a Note Transfer History;

Figure 17 illustrates a Note Transfer Tree;

Figures 18A-18C illustrates a Link Money Module to Bank Account(s) for Bank Access protocol;

Figures 19A-19C illustrate a Revalidate Money Module Link to Bank Account(s) protocol;

Figure 20 illustrates a Valid Account Number protocol;

Figures 21A-21B illustrate a Create Lost Notes claim protocol; and

Figures 22A-22E illustrate a Claim Lost Notes protocol.

DETAILED DESCRIPTION OF THE INVENTION

The present invention provides an improved monetary system using electronic media to securely and reliably exchange economic value. The system can be implemented by integrating novel data processing systems with other procedures which can be implemented with the current worldwide banking systems. In accordance with the present invention, several types of enhancements are presented to the Electronic Monetary System, disclosed in commonly assigned, U.S. Patent Application Serial No. 07/794,112, filed November 15, 1991, which is herein incorporated by reference. These enhancements include: a set of security improvements including improved foreign exchange (F/X) and Transfer Notes transaction processes, a process to claim lost money, a process for linking money

AUSTRALIA PATENT OFFICE

[THIS PAGE IS INTENTIONALLY BLANK]

modules for bank access, a process for a point of sale payment, and a process for updating credit.

SECURITY

Effective security for a monetary system has three characteristics: inhibit counterfeiters, detect counterfeiters, and contain counterfeiters. The described EMS is designed to have components which exhibit all three characteristics.

In order to inhibit counterfeiters, the money modules communicate using symmetric and asymmetric key cryptography. None of the messages are in the clear. The module's protocols are also physically protected by tamper-proof hardware.

Counterfeiting is detected by note reconciliation processes. System-wide time protocols (e.g., note expiration) force electronic notes to be reconciled regularly. Electronic notes are also refreshed (i.e., replaced with a new note with a new expiration date) when banking transactions are performed.

Money modules are blocked (e.g., placed on the bad ID list) if duplicated or counterfeit notes are tied back to them. Also, notes which have passed through these modules will not be allowed to transfer. The transfer of duplicated or counterfeit notes will be contained since notes expire or eventually are deposited to a bank. Moreover, in case of a serious system security problem, the EMS may call for a global recertification, thereby requiring all modules to recertify, including transaction money modules the next time they sign on the EMS Network.

The following list of improvements are intended to shield all information from eavesdroppers listening to module interactions. All information is cloaked--even the public keys and module identifiers. The list of improved security features include the following.

1) Securing the network sign-on so that no one can spoof the money module or intercept any

of its information in the clear, as described below with reference to Figure 5.

2) Creating a method for assigning security server, money generator and teller identifiers (see Module Numbering Scheme). These identifiers are checked in:
   a) Establishing a session (see Figure 7);
   b) Transferring notes--check transfer records (see Figure 8).

3) Implementing a two-tier security server structure (see Security Hierarchy and Security Network), including a primary security server, wherein all modules carry the public keys of the primary security server, and a security server which certifies all other modules.

4) Improving Transfer Notes to check bad ID list against all transfers before accepting notes in payment or Foreign Exchange (F/X), and to check for duplicated notes (see Figure 8).

5) Encrypting all certificates using private keys of a security server (see Certificate Structure and Validation).

6) Varying the size of the public keys dynamically (see Security Network and Figure 6).

7) Changing the commit protocol so that failure does not duplicate money (see Figure 10).

8) Changing F/X so that neither party can interrupt the commit, in order to cheat with certainty, by receiving money and not sending any money (see Figure 9).

9) Changing Abort to log information on a failed commit if payer commits but receiver

aborts (see Figure 11).

10) Allowing global recertification if needed (see Security Network and Figure 6).

The foregoing list of security improvements highlights some of the security features provided by the enhanced security system of the present invention. These, and other, improvements may be further understood from the ensuing detailed description of an alternative preferred embodiment of a security system for an EMS.

Security Hierarchy

In accordance with the present invention, another embodiment for EMS system security may be provided as follows. Referring to Figure 1A, EMS will have two types of security servers, primary 1182 and ordinary 1184. The primary security servers 1182 certify the (ordinary) security servers 1184. The security servers 1184 certify all other modules (transaction MMs 1186, Teller MMs 1188, money generator modules 1190, and customer service modules 1192) in the system.

The primary servers 1182 only interact with other primary servers 1182 or the security servers 1184. Referring to Figure 2, the primary security servers 1182 are housed in a secure facility connected to each other by a Security LAN 1194. The LAN 1194 is connected through a secure gateway to the Security Network 1196. Only the security servers communicate over this network. All security servers are physically protected devices.

Security servers 1184 are also attached to the EMS Network 1198 and bank local networks 1200. Security servers are treated as if they could be compromised and are validated upon all interactions with other modules.

Only the security servers 1184 and modules have certificates. The primary security server's public keys are carried by these devices. There are two types of certificate: security server and module.

Certificate Structure And Validation

The structure of a certificate is as follows:

$$\text{Cert}(SS)=E_{PSS}[\underbrace{SS(id)\,\|\,SS(PK)\,\|\,\text{expire date}}_{X}\,\|\,\sigma_{PSS}(X)]\,\|\,[PSS(id)\ \text{XOR}\ C]$$

$$\text{Cert}(M)=E_{SS}[\underbrace{M(id)\,\|\,M(PK)\,\|\,\text{expire date}}_{Y}\,\|\,\sigma_{SS}(Y)]\,\|\,\text{Cert}(SS)$$

The certificate validation protocols are:

1) Validate Cert(SS)

   a) $PSS(id) = [PSS(id)\ \text{XOR}\ C]\ \text{XOR}\ C$

   b) $D_{PSS}(E_{PSS}(X\,\|\,\sigma_{PSS}(X)))=X\,\|\,\sigma_{PSS}(X)$

   c) Check if SS(id) is authentic (see module numbering scheme)

   d) Check if date is valid

   e) Check if $D_{PSS}(\sigma_{PSS}(X))=h(X)$

2) Validate Cert(M)

   a) Validate Cert(SS)

   b) $D_{SS}(E_{SS}(Y\,\|\,\sigma_{SS}(Y)))=Y\,\|\,\sigma_{SS}(Y)$

   c) Check if M(id) is authentic (see module numbering scheme)

   d) Check if date is valid

   e) Check if $D_{SS}(\sigma_{SS}(Y))=h(Y)$

Where PSS=Primary Security Server

SS=Security Server

M=Module

$\|$=Concatenate

id=identification number

h=Hash function

C=Constant random number

PK=Public Key (includes length of key)

σ=Digital Signature=$E \circ h$

Cert=Certificate

E=Algorithm with private key used for encrypting and for creating digital

shared by all modules	signature D=Algorithm with public key used for decryption and for checking digital signature

Note E and D may also be used for decrypting and encrypting, respectively, when applied in other applications.

Module Numbering Scheme

The primary security servers 1182, security servers 1184, teller money modules 1188, money generator modules 1190, customer service modules 1192, and transaction money modules 1186 are assigned identification numbers (id's) so that the numbers can be checked for authenticity. A 48-bit prime number "p" is generated and a primitive root "a" modulo p (where $a^n \not\equiv 1(p)$ for all $1 \le n < p-1$) is found via a secure process. Both a and p are loaded to all modules in the system securely by the primary security servers when they are manufactured.

The scheme works as follows:

If $a^n \equiv m(p)$ and

(1) $1 \le m \le 99{,}999$ then n is assigned as the id of a primary security server,

(2) $100{,}000 \le m \le 999{,}999$ then n is assigned as the id of a security server,

(3) $1{,}000{,}000 \le m \le 6{,}999{,}999$ then n is assigned as the id of a teller money module,

(4) $7{,}000{,}000 \le m \le 9{,}999{,}999$ then n is assigned as the id of a money generator module,

(5) $10{,}000{,}000 \le m \le 11{,}999{,}999$ then n is assigned as the id of a customer service module,

(6) $m \ge 12{,}000{,}000$ then n is assigned as the id of a transaction money module.

If a module or server is validating a certificate, it checks the authenticity of the identification number (e.g., M(id), SS(id), or PSS(id)) n by computing $a^n \equiv m(p)$ and then checks if m is in the correct range.

Security Network

As shown in Figure 2, the Security Network 1196 and the Security LAN 1194 connect the security servers 1184 to the primary security servers 1182. Security servers 1184 initially certify the money modules and customer service modules 1192 at manufacturing. Such security servers may be connected by a Module Manufacturing LAN 1202. They pass security information such as the bad id list and the list of primary security servers and their public keys to the modules. The bad id list contains the identities of the money modules, customer service modules, and security servers which are blocked from transacting. Recertification of these modules is described subsequently in the network sign-on flow diagram.

The security servers 1184 are initially certified by the primary security servers 1182 at manufacturing. Such primary security servers may be connected by a Security Server Manufacturing LAN 1204. Referring to Figure 1B, the security servers 1184 receive various security information which they pass to the other modules. The security servers provide security services for the EMS Network 1198 and the bank LANs 1200, such as network sign-on where they pass updated security information. The security servers 1184 receive this information from the primary security servers 1182 over the Security Network 1196. Transaction money modules 1186 communicate with the EMS Network 1198 via network servers 1206 (NS). Participating banks have teller money module(s) 1188 and perhaps money generator(s) 1190 connected to their LANs 1200.

The Security Network 1196 is link encrypted. In addition, the primary security servers and security servers share a common symmetric key (a Security Network encryption key). This key is changed periodically by a designated primary server 1182 by public key, key exchange.

The list of bad id's is maintained by a designated primary server 1182. The list is accumulated from interactions with participating banks, law enforcement authorities, and subscribers to the system.

Periodically the length of the public keys for the security servers and the modules will be changed. The key length will be normally lengthened to maintain a high security level. The new designated key lengths will be communicated to the primary security servers by a designated primary server. The new lengths will be communicated to the security servers by the primary servers when new bad id lists are sent or upon recertification. In case of a dangerous breach of security, a primary security server can call for global recertification.

The length of the public key for each primary server will not change. A timetable will be created which will schedule the implementation and decommission of primary security servers. The new servers will most likely have longer keys unless they are implemented because of increased transaction volume. The list of active PSS public keys is created by a primary security server and signed by the server with its private key. The list is then broadcast to other security servers.

Figure 3A shows the functional components of a security server 1184. An External Interface function 1208 provides a communications layer for network interfacing. A Session Manager function 1210 controls the security aspects of a transaction session. A Network Sign-On function 1212 manages the security functions for network sign-on. A Create Certificate function 1214 certifies a certificate for any of the money modules (in a primary security server, this function certifies security servers). A Distribute Certificatory Keys function 1218 distributes the Certification Agency's list of valid primary security server public keys to the money modules (primary security server also distributes global recertification message). A Control Bad ID List function 1220 controls and distributes the list of bad identifiers. A Synchronize Date/Time function 1222 keeps money module Clock/Timer services synchronized to a system time. Clock/Timer 1224 and Cryptography functions 1226 are identical to those functions in the money modules.

Figure 3B shows the functional components of a network server 1206. An External Interface function 1228 provides a communications layer for network interfacing. A Communication Session manager function 1230 manages a communication session between money modules, and between a money module and a security server. A Network Sign-On function 1232 controls the money module network sign-on process. A Route Message function 1234 provides directory services for routing messages, controlling message routing during sign-on and during a money module session. A Direct to Bank Services function 1236 provides information on services provided by participating banks. A Cryptography function 1238 provides a Symmetric Key function 1240 and a Random Number Generator function 1242. The Symmetric Key function 1240 encrypts messages between the network server 1206 and the modules accessing the network and between the network server 1206 and the security servers 1184. The Random Number Generator function 1242 generates random numbers for encryption keys and verification messages.

In accordance with the present invention, another secure processing component, a customer service module (CSM) 1192, is preferably employed. A CSM is a tamper-proof device used for creating and updating account profiles. CSMs contain a unique certificate like those found in money modules and security servers. CSMs can establish secure sessions with other modules (e.g., security servers). The CSM requires a host to interface to a customer service representative and the on-line banking systems.

The CSM has two primary functions. First, the CSM creates account profiles so that a money module can access bank accounts, revalidate the money module link to bank accounts, and validate account numbers. These transactions are more fully described hereinbelow with reference to figures 18-20. Second, the CSM functions to claim lost notes in response to a request from a host customer services representative, which is described in further detail in figure 21 and figure 22. A CSM has the same security functions as a money module and a special range of numbers for its identifier (see "<u>Module Numbering Scheme</u>"). The performance of these functions by the CSM simplifies the validate account number process for the Teller Module.

In accordance with an embodiment of an EMS which employs a CSM, the account profile structure for each bank changes to:

$$\underbrace{\text{Expire date} \,\|\, \text{M(id)} \,\|\, \text{B(id)} \,\|\, \text{LA}}_{X} \,\|\, \sigma_{csm}(X) \,\|\, \text{Cert(CSM)}$$

Where M(id) = module identifier
B(id) = bank identifier
LA = list of account numbers and type of account (deposit or loan)
σ_{csm} = signature of CSM
Cert(CSM) = certificate of (CSM)
$\|$ = concatenate

A validation procedure for such a profile is described hereinbelow with reference to figure 20.

Figure 4A shows the functional components of a Customer Service Module (CSM) 1192. The External Interface 3000 interfaces the CSM to other processing and communications means within the Customer Service Module

Host (CSMH) processor; the Session Manager 3001 acts to control and commit (i.e., finalize) or abort a transaction session between the CSM and a transaction money module or customer service representative. A Create Account Profile function 3002 constructs from customer account information an account profile that allows a money module to access the subscriber's different bank accounts. A Public Key function certifies and signs a bank account profile. Since the CSM requires a host to interface to a customer service representative and the on-line banking systems, a To Host function 3006 mediates the separation of duties between the CSM applications and the host applications. A Claim Lost Notes function 3008 is responsive to subscriber lost note claims, which the CSM validates and distributes to the issuing banks. The Maintain Security function 3004 manages the list of compromised money modules, applies for certificates, synchronizes the clocks, and manages the creation of new digital keys. Clock/Timer 3012 and Cryptography functions 3010 are identical to those functions in the money modules.

Figure 4B shows the functional components of a primary security server 1182. An External Interface function 3020 provides a communications layer for network interfacing. A Session Manager function 3002 controls the security aspects of a session with security servers, which are treated as if they may be compromised, and with other primary security servers. A Create Certificate function 3024 creates a certificate for any of the security servers. A Distribute Certificatory Keys function 3026 distributes the Certification Agency's list of valid primary security server public keys to the security servers. A Distribute Security Network Keys function 3032 manages and distributes security network keys among primary security servers, and to security servers. A Set Global Recertification function 3030 determines whether global recertification is required (e.g., because of a dangerous breach of security) and calls for global recertification when deemed necessary. A Distribute Bad ID List function 3028 controls and distributes the list of bad identifiers. Clock/Timer 3034 and Cryptography functions 3036 are identical to those functions in the money modules.

Network Sign-On

An overview of the network sign-on procedure is provided with reference to Figure 5. The Sign-On protocol describes the situation where a module 1243 desires access to the EMS Network 1198 for recertification, deposit, withdrawal or other reason. The module 1243 may be a transaction money module 1186, teller money module 1138, money generator module 1188, or customer service module 1192. (a) Establish a communication between module 1243 and network server 1206. (b) Pass the module's certificate to the network server 1206. (c) The network server 1206 generates a random verification number V and a random key K; the network server then passes the module's certificate, V, and K to a security server 1184 (encrypted by a NS/SS key). (d) The module 1243 and the security server 1184 establish a secure communication session (via session key (MM/SS)). (e) The security server 1184 passes the time/date, update bad ID list, update list of primary security server public keys, public key length, global recertification (if necessary), and recertified module certificate (if necessary). (f) End session with module 1243 and send V and K to the module 1243. (g) Encrypt V with K and send to the network server 1206. (h) The network server 1206 acknowledges network sign-on to the module 1243. (i) The module 1243 then informs the network server 1206 of the destination (if any) to which it wishes to be connected. (j) The network server 1206 establishes a connection to the destination.

The network sign-on is designed so that no one can spoof the module 1243 or intercept any of its information in the clear. Figure 6 describes the detailed flow of the network sign-on procedure.

Communications A establishes communications with the EMS Network 1198 (step 1244). Maintain Security A sends its certificate to the network server 1206 (step 1246). NS Network Sign-On receives the certificate (step 1248). NS Random Number Generator generates random key K and random verification number V (step 1250). NS Symmetric Key encrypts the module's certificate, K and V with a NS/SS key (step 1252). NS/SS keys are local symmetric keys installed in network servers 1206 and security servers 1184 which communicate for network sign-on. NS Network Sign-On sends the certificate, K and V to the security server 1184, where SS Network Sign-On receives the message and SS Symmetric Key decrypts the message (steps 1254 - 1258). SS Network Sign-On stores K and V and then sends the module certificate to SS Public Key for validation (steps 1260 - 1264). In order to allow for the possibility of recertification, SS Public Key does not consider the expiration date in determining the validity of the module certificate.

If the module certificate is not valid, then SS Network Sign-On creates messages to deny access for transmittal to the network server 1206 and module 1243 (step 1266). SS Public Key encrypts the message to the module 1243 with the module's public key and SS Session Manager sends the messages to the network server (step 1268 - 1270). NS Network Sign-On receives the messages and notes access denied. The encrypted message is then sent to the module, and the network server disconnects. (Step 1272). Session Manager A receives the message, Public Key A decrypts the message, and Session Manager A notes that sign-on was denied (steps 1274 - 1278). If the device requesting sign-on was a transaction money module, then To Subscriber A informs the subscriber (steps 1280 - 1282). Otherwise, To Bank A informs the bank (step 1284).

If, on the other hand, the module's certificate is valid, then SS Control Bad ID List checks if the module's id is on the bad id list (steps 1286 - 1288). If the id is on the list, then network access is denied. Otherwise, SS Random Number Generator creates random number R and a verification message (step 1290). SS Network Sign-On assembles R and the verification message in a message which is encrypted by SS Public Key using A's public key. Public Key A also augments this encrypted message by appending the security server's certificate. (steps 1292 - 1294). The message is sent to A where Public Key A decrypts the message and validates the security server's certificate (step 1298).

If the certificate is invalid, then A notes session termination and informs either the subscriber or the bank (steps 1304 - 1306). If the certificate is valid, then Maintain Security A checks if the security server's id is on the bad id list (steps 1308 - 1310). If on the list, then the session is terminated (steps 1300 - 1306). If not on the list, then Random Number Generator A creates random number R(A) (step 1312). Maintain Security A forms session key (MM/SS) by the operation R(A) XOR R and then stores the session key (step 1314).

A message containing the verification message and R(A) is assembled and encrypted with the security server's public key (step 1316). Session Manager A sends the message To SS Network Sign-On and SS Public Key decrypts the message (steps 1318 - 1322).

SS Network Sign-On verifies that the verification message is the one which it created (steps 1324 - 1326). If it is not, then the security server denies network access. If the verification message is correct, then SS Symmetric Key forms session key (MM/SS) by R(A) XOR R (step 1328). SS Session Manager notes the start of session and sends an acknowledgement to A by the Send Message subroutine (steps 1330 - 1332). Session

Manager A receives the acknowledgement and notes the start of the session (step 1334).

Clock/Timer A sends the time and date to the Session Manager which sends it to the security server (steps 1336 - 1340). SS Synchronize Date/Time receives the time and date and checks if it is within parameter (steps 1342 - 1344). If not within parameter, then SS Synchronize Date/Time sends a new time and date to Session Manager A (steps 1346 - 1350). Clock/Timer A then adjusts the time and date (step 1352). A then resends its date and time to the security server for rechecking. If clock synchronization is attempted more than a set number of times, then a clock malfunction is reported to the subscriber or bank, which may then retry if desired (steps 1354 - 1362).

If, however, the time and date are within parameter, then SS Network Sign-On assembles a message containing the bad id list, the new list of primary security server public keys (which comes from the Distribute Certificatory Key function), and the public key length (the size of the public keys are varied periodically) (step 1364). SS Create Certificate checks if a global recertification has been called for, and ensures that the time period for the global recertification has not expired (steps 1366 - 1368). Such time period should be long enough so that everyone's certificate has been recertified or expired. The function should also check when the module last recertified because if it was certified during the period of the global recertification, then there would be no need to recertify.

If recertification is required, then SS Create Certificate adds to the previous message: module should recertify (step 1370). Then, whether or not a recertification is called for, SS Public Key signs the message (step 1372). The message is sent to A where Public Key A checks the digital signature on the message (steps 1374 - 1378). If the signature is invalid, then the session is terminated. Maintain Security A then updates its bad id list, public key list, and key length (step 1382).

Module A then checks if its certificate needs to be recertified (either because of a global recertification order or because of an expired certificate) (steps 1384 - 1386). If a new certificate is required, then Maintain Security A initiates the generation of a new certificate (step 1388). Public Key A generates new keys and signs the new public key with its old private key (step 1390). Session Manager A sends the signed new public key to the security server's SS Create Certificate (steps 1392 - 1396). SS Public Key then validates the signature on the new public key (steps 1398 - 1400). If not a valid signature, then the security server denies network access. If the signature is valid, then SS Public Key signs the module's new certificate and sends it to the module (step 1402). Session Manager A receives the certificate, Maintain Security A undertakes to validate the certificate, and Public Key A validates the signature (steps 1404 - 1410).

If the certificate is not valid, then Session Manager A sends a "Certificate Invalid" message and the certificate to the security server by the Send Message function (steps 1412 - 1413). SS Network Sign-On receives the message and SS Public Key validates the signature (steps 1414 - 1418). If the security server determines that the certificate is actually valid, then it denies the module access to the network. If, however, the certificate is invalid, then SS Session Manager informs the network server that it will disconnect from the network (step 1420). NS Network Sign-On informs the module of the malfunction (step 1422). The module then queries the subscriber or the bank for a retry (steps 1424 - 1432).

If, on the other hand, the module determines that its new certificate is valid, then Session Manager A sends an acknowledgement to the security server (step 1434). Similarly, if no new certificate was required then Maintain Security A sends an acknowledgement message to the security server (steps 1436 - 1438). In either case, SS Session Manager receives the acknowledgement and notes the end of its session with the module (step 1440). SS Network Sign-On then sends K and V to A (steps 1442 - 1444). Session Manager A receives the message and Symmetric Key A encrypts V and the destination with K and sends the message to the network server (steps 1446 - 1448). NS Network Sign-On receives the message and NS Symmetric Key decrypts the message and checks if V is the same V it previously generated (steps 1450 - 1454).

If V is incorrect, then NS Network Sign-On sends an access denied message to A and then disconnects (steps 1456 - 1458). If V is correct, then NS Network Sign-On establishes a link to the destination and sends an acknowledgement to A (step 1460). Finally, Session Manager A receives the acknowledgment and notes that A has signed-on to the EMS Network 1198 (step 1462).

Establish Session

Figure 7 shows the Establish Session protocol. Maintain Security A sends the module certificate to the session manager, and Session Manager A receives the certificate and checks if money module A is connected to the network (steps 1464 - 1466). If money module A is not connected to the network, then Session Manager A sends the certificate received from Maintain Security A to destination B (step 1476).

Alternatively, if money module A is connected to the network, then Symmetric Key A encrypts the certificate with K and Session Manager A sends the encrypted certificate to the network server (step 1468 - 1472). The Network Server decrypts the certificate with K and sends the certificate to destination B.

Regardless of whether the certificate was sent by the Network Server or by Session Manager A, Session Manager B receives the certificate and Maintain Security B (if B is a security server, then this function is performed by the Session Manager) validates the certificate (steps 1480 - 1482). If the certificate is not valid, then Session Manager B notes the session is terminated and informs either the subscriber or the bank (steps 1486 - 1492) (if B is a security server, then B merely notes the transaction is terminated).

If the certificate is valid, then Maintain Security B checks if A is on the bad id list (steps 1494 - 1496). If A is on the list, then the session is terminated. If A is not on the list, then Random Number Generator B creates random number R(B) and a B verification message (step 1498). Clock/Timer B retrieves the time and date (step 1500). Maintain Security B assembles R(B), B verification message and time and date in a message (step 1502). Public Key B encrypts the message with A's public key and Session Manager B appends B's certificate to the encrypted message and sends the message to A (steps 1504 - 1506).

Session Manager A receives the message, Public Key A decrypts the encrypted part of the message, and Maintain Security A validates B's certificate (steps 1508 - 1514). If the certificate is not valid, then Session Manager A notes the termination of the session and informs either the subscriber or the bank (steps 1516 - 1522). If the certificate is valid, then Maintain Security A checks if B is on the bad id list (steps 1524 - 1526). If B is on the list, then the session is terminated. If B is not on the list, then Maintain Security A retrieves the date and time and compares it to B's date and time (steps 1528 - 1530). If the date and time are out of range, then the session is terminated.

If the date and time are in range, then Random Number Generator A creates random number R(A) and an A verification message (step 1532). Maintain Security A then forms a session key by the operation R(A) XOR R(B) (step 1534). The A verification message, the B verification message, the time, date and R(A) are assembled into a message and encrypted with B's public key (step 1536). The message is sent to B by Session Manager A (step 1538). Session Manager B receives the message, Public Key B decrypts the message and Maintain Security B checks the B verification message (steps 1540 - 1546). If the B verification message is incorrect, the session is terminated. If the B verification message is correct, then Maintain Security B forms the session key by R(A) XOR R(B) (step 1548). The time and date are retrieved and compared to A's time and date to check if they are within a predefined range of each other (step 1550). If the time and date are out of range, then the session is terminated. If the time and date are in range, then Session manager B notes the start of the session (step 1552).

Session Manager B then sends an acknowledgement and the A verification message to A (steps 1554 - 1556). Session Manager A receives the message and Maintain Security A checks the A verification message (steps 1558 - 1562). If the verification message is not correct, the session is terminated. If the verification message is correct, then Session Manager A notes the start of the session (step 1564).

Transfer Notes

Figure 8 shows the transfer notes protocol. Note Directory X chooses the note(s) and values for transfer, updates the note amount(s) and sequence number(s), and then sends the message to Notes (step 1566). Possible objectives in choosing the notes for transfer may, for example, be: (1) minimize the number of digital signatures (which requires processing time); (2)

minimize the size of the packet; (3) maximize the usefulness of the electronic notes left to the transferring subscriber (i.e., pass the notes with the shortest time left until expiration). Such objectives may be achieved by the following note transfer algorithm: (1) determine all possible alternatives which contain the least number of notes; (2) determine which of these alternatives have the least number of transfers; (3) if more than one choice is left from step 2, choose the one which has the least number of monetary unit days. Monetary-unit days = residual value of note to be transferred times the number of days left until the note expires, summed for all the notes in the packet.

Notes X, upon receiving the message from Note Directory X, creates a transfer to be appended to each note being transferred (step 1568). Public Key X creates signatures for the note(s) (step 1570). Packet Manager X then assembles the note(s) with their new transfer(s) and signature(s) in a packet and sends the packet to Y (steps 1572 - 1574). Packet Manager Y receives the packet and disassembles it (step 1576).

Verify Y validates all certificates in the note(s) (e.g., money generator certificate and all transfer certificates). Then Verify Y verifies that the identification numbers in the transfer group match up with the module identification numbers of the certificates in the signature and certificate group throughout the history of the electronic note. Also, the consistency of the transfer amounts for each note is validated by ensuring that throughout the electronic note history the amount transferred in each successive transfer is less than that of the immediately precedent transfer. In addition, the total amount transferred is checked to ensure it is the amount expected. (Steps 1578 - 1580). If not valid, then the transaction is aborted (step 1582).

If valid and Y is a transaction money module, then Verifier Y verifies the expiration dates of the note(s) (steps 1584 - 1588). If any of the note(s) have expired, then the transaction is aborted. If none have expired, then Verifier Y checks each id from the note transfers against the bad id list (steps 1590 - 1592). If any of the transfer id's are on the bad id list, then the transaction is aborted.

If the transfer id's are not on the bad id list (or Y is not a transaction money module), then Public Key Y verifies the validity of the note(s) signatures (steps 1594 - 1596). If any of the signatures are not valid, then the transaction is aborted. If the signatures are valid, then Verifier Y checks if the note(s) bodies match note bodies that are stored by the Notes application or located in the Tran Log (steps 1598 - 1600). For each note body that matches, a note transfer tree is created in order to determine whether there has been any note duplication (steps 1602 - 1604). If any of the notes have been duplicated, then the transaction is aborted. This check for duplication (i.e., steps 1598-1604) is particularly directed to, and well suited for, thwarting individuals who attempt to create money by transferring notes by "self-dealing" using a compromised transaction money module.

If there are no duplicates, or if no matches of note bodies were identified, then Notes Y places the note(s) in the money holder (step 1606). Finally, Note Directory Y updates the note(s) location(s) and amount(s), and also initializes sequence number(s) (step 1608).

It may be understood that the Transfer Notes process includes steps for updating and initializing a sequence number to facilitate note reconciliation (see "<u>Money Issued Reconciliation</u>"), checking if the transferee of any note is on the bad id list, and checking for note duplication. These additional features and steps make it difficult for adversaries to introduce and circulate duplicated notes, and enhance the ability to detect duplicated notes in circulation.

Foreign Exchange

Figure 9 shows the protocol for a foreign exchange transaction using dollars and pounds as exemplary monetary units. Initially, A agrees to exchange with B dollars($) for pounds (£) at an exchange rate of $/£ (step 1602). A and B then sign on their money modules and the modules prompt their subscribers for the type of transaction (steps 1604 - 1610). A chooses to buy foreign exchange and B chooses to sell foreign exchange (steps 1612 - 1614). A and B establish a secure transaction session (steps 1616 - 1620).

To Subscriber A prompts the owner/holder of A for the amount by type of note of dollars he wishes to exchange (step 1622). Pay/Exchange A receives the amount and Note Directory A checks if A has sufficient funds (steps 1624 - 1628). If the funds are not sufficient, then To Subscriber A prompts for a new amount which again is checked against existing funds (steps 1630 - 1632). If no new amount is entered, then the transaction is aborted (step 1634).

If funds are sufficient, then Pay/Exchange A sends the dollar amount to B (steps 1636 - 1638). To Subscriber B then prompts the owner/holder of B to select either the amount of pounds he wishes to exchange for the dollars or, alternatively, simply the exchange rate for the dollars (step 1640). Note Directory B checks for sufficient funds (steps 1642 - 1644). If funds are insufficient, then To Subscriber B prompts for a new rate and again existing funds are checked for sufficiency (steps 1646 - 1648). If, however, no new rate is selected, then Pay/Exchange B informs A of its insufficient funds (steps 1650 - 1652). A may then select a new amount for exchange or abort the transaction (steps 1630 - 1634).

If B has sufficient funds for the transaction, then Pay/Exchange B sends A an acknowledgement and the amount of pounds to be exchanged (the equivalent rate is also sent) (steps 1654 - 1656). To Subscriber A prompts to verify the amount of pounds and the rate (steps 1658 - 1660). If the amount and rate are incorrect, then Pay/Exchange A informs B that the amount and rate are incorrect (steps 1662 - 1664). To Subscriber B then prompts for a new rate (steps 1666 - 1668). If no new rate is chosen, then the transaction is aborted (step 1670).

If, however, A verifies the correct amount and rate of the transaction, then Pay/Exchange A passes the dollar amount to the money holder (step 1672). The dollar notes are then transferred from A to B (step 1674). Pay/Exchange B passes the pound amount to its money holder (step 1676). The pound notes are then transferred from B to A (step 1678).

At this point in the transaction, both A and B provisionally hold foreign exchange notes in the correct amounts. A and B have each participated in two transfers: A transfers: (1) A transferred dollars to B; (2) A received pounds from B. B transfers: (1) B transferred pounds to A; (2) B received dollars from A. To complete the foreign exchange transaction, A must now commit (i.e., finalize and permanently record in its transaction log) both of its two transfers. Similarly, B must commit both of its two transfers. Note that A may commit to the foreign exchange transfer A → B (dollars from A to B) and B → A (pounds from B to A) separately. Likewise B may commit to the foreign exchange transfers A → B and B → A separately.

The next portion of the foreign exchange protocol is designed so that neither party knows the order in which the transacting money modules will commit. Such uncertainty will discourage parties from intentionally trying to tamper with a transaction. As background, a function S(X) is defined so that S(0) = A and S(1) = B, where A and B refer to money modules A and B. Thus, if X is randomly chosen as 0 or 1, then money module A or B is randomly indicated.

The following routine is used to allow A and B to commonly establish a random X. R(A) and R(B) are the random numbers generated by A and B, respectively, during the Establish Session subroutine. The parity of R(A) XOR R(B) is determined (by exclusive - ORing each bit of R(A) XOR R(B). This parity is the random number X. $\overline{X}$ is the complement of X ($\overline{X}$ = X XOR 1).

Referring again to Figure 9, Tran Log A conditionally updates its transaction log to record the transfer S(X) to S($\overline{X}$) (step 1680). If X is calculated to be 0, then the transfer A to B (i.e., the dollar transfer) is conditionally recorded. If X is calculated to be 1, then the transfer B to A (i.e., the pound transfer) is conditionally recorded. Because the log is conditionally recorded, it may be rolled-back in the event money module A aborts the transaction. The update log becomes permanent once the log update has been set to unconditional (either as shown explicitly in the flow diagram or implicitly during a commit). Session Manager A then sends a "Log Updated" message to B (steps 1682 - 1684). In response, Tran Log B also conditionally updates its log to record the transfer S(X) to S($\overline{X}$) (step 1686).

If X = 1, then Tran Log B sets the log update to unconditional (steps 1688 - 1690). Thus, at this point, B has committed to its transfer of pounds to A. Next, B follows the Commit protocol (step 1692) described subsequently with reference to Figure 10. In this situation, A will commit to both its transfers (i.e., transferring dollars and receiving pounds) and B will commit to its one outstanding (uncommitted) transfer, namely receiving dollars.

If, however, X = O (step 1688), then Session Manager B sends a "Start Commit" message to A (steps 1694 - 1696). Tran Log A then sets its log update to unconditional (step 1698), thus committing to its transfer of dollars. The Commit protocol of Figure 10 is then called (step 1700). During this protocol (described subsequently) B commits to both its transfers (i.e., transferring pounds and receiving dollars) and A commits to its one outstanding transfer, receiving pounds.

As can be seen, the F/X transaction commits each transfer of notes, thus keeping notes from being duplicated, which could occur if there were only one commit. The foreign exchange protocol ensures that neither party knows whose transfer (A's of dollars or B's of pounds) will be committed first. This reduces the incentive of a party to tamper with the transaction. Each transfer is committed separately, but the order of commit is chosen randomly by the two modules. A party who tries to interfere with the transaction has a 50-50 chance of losing money. Also, there is no incentive to interfere with the commit since the lost money can be reclaimed (see Claim Lost Money).

Commit

Figure 10 shows the Commit protocol for modules. Session Manager X sends a "Ready-to-Commit" message to Y (steps 1702 - 1704). This passes the obligation to commit to the module receiving the message. In a conventional money transfer scenario, this technique of passing the burden of committing first is used to ensure that the party transferring money commits first, so as to eliminate the possibility of duplicating money.

Session Manager Y then sends an acknowledgment to X (steps 1706 - 1708) and commits to any outstanding transactions by updating its transaction log (step 1710). Also, if Y is a transaction money module, then To subscriber Y notifies the subscriber of the successful transaction (steps 1712 - 1714). Session Manager Y notes the end of the session (step 1716).

Tran Log X receives the acknowledgement from Y and updates its transaction log, thus committing to any outstanding transfers. X completes its commit in the same manner as Y (steps 1718 - 1724).

Abort Transaction

Figure 11 shows the Abort transaction protocol for modules. Session Manager X rolls-back changes and notes that the transaction is aborted (step 1726). Session Manager X then checks if the "Ready-to-Commit" message has been sent (steps 1728 - 1730). If so, then X updates its transaction log (step 1732) by recording that X aborted after sending a Ready-to-Commit message and recording the note identifiers and amounts of each note received during the Transfer Notes protocol. Thus, the abort protocol logs information when the Abort subroutine is called during a failed Commit subroutine.

If X is a transaction money module 1186, and the Ready-to-Commit message was sent, then To Subscriber X informs its subscriber that the transaction was aborted and that there may have been a money transfer error (steps 1734 - 1738).

If X is a teller money module 1188, then To Bank X informs the bank that it should reverse its accounting transactions (by appropriate debits and credits) (steps 1740 - 1742). If X is a transaction money module 1186 and no Ready-to-Commit message has been sent, then To Subscriber X informs the subscriber that the transaction was aborted (step 1744).

In any event, Session Manager X then sends Y a message that the transaction cannot be completed (steps 1746 - 1748). Session Manager Y rolls-back its changes and notes the transaction as aborted (step 1750). Y then informs its subscriber that the transaction is aborted (steps 1752 - 1754) or informs the bank to reverse its accounting transaction (steps 1756 - 1758).

As described, if a transaction is interrupted during a commit protocol, it is possible that notes will be lost. If this occurs, the transferee will have aborted and the transferor will have committed to the transfer of notes. In this case, the transferee money module records information about the notes it should have received and notifies the subscriber that there is a potential problem (i.e, it did not receive the notes sent by A). It may be noted that in this circumstance, as far as the transferor money module is concerned, it properly transferred the notes.

The transferee money module subscriber can then make a claim for the money to the Certification Agency. The claim information would include the log record of the failed transaction. The Certification Agency could then check with issuing banks to see if the notes have been reconciled. After some period of time, if the notes have not been reconciled, the subscriber could reclaim his money.

POS Payment

Figure 12 shows a Point of Sale (POS) payment protocol. The POS Payment protocol is intended to simplify payments made between a buyer's transaction money module 1186 and a merchant's transaction money module 1186. The merchant's transaction money module 1186 may, for example, be located in a cash register at a supermarket.

Initially, A agrees to purchase products or services from B (step 1760). The owner/holder of transaction money module A signs onto his money module (step 1762). To Subscriber A prompts the owner/holder for a transaction and A chooses to make a POS payment (steps 1764 - 1766). Meanwhile, the merchant determines the total purchase price (step 1768). To Subscriber B prompts for a transaction and B chooses to receive a POS payment (steps 1770 - 1772). A and B then establish a secure session (steps 1774 - 1776).

To Subscriber B prompts for amount of payment and Pay/Exchange B receives the amount and sends it to A (steps 1778 - 1782). To Subscriber A then prompts its subscriber to verify the requested amount (steps 1784 - 1786). Moreover, the subscriber is requested to choose the notes in which it will pay (e.g., currency or credit) and the amounts so that the total equals the requested amount. If the requested amount is not correct, then Pay/Exchange A sends B a message indicating that the requested amount is incorrect (steps 1788 - 1790). To Subscriber B then prompts its host for a new amount (steps 1792 - 1794). If a new amount is not chosen then the transaction is aborted (step 1796).

If the requested amount is correct, then Pay/Exchange A receives amounts by type of note (step 1798). Note Directory A then checks for sufficient funds (steps 1800 - 1802). If funds are insufficient, then To Subscriber A prompts for new amounts by type of note (steps 1804 - 1806). If no new amount is entered, then Pay/Exchange A sends B a message that it has insufficient funds (steps 1808, 1790). To Subscriber B prompts host for new amount (steps 1792 -1794). If no new amount is selected, then the transaction is aborted (step 1796). If a new amount is selected, then the payment transaction begins again.

If funds are sufficient, then Pay/Exchange A passes the amount to the money holder (step 1810). The notes are then transferred from A to B (step 1812). Finally, the transaction money modules commit (step 1814).

As can be seen, the POS payment is simplified for the buyer because it is a payee initiated payment. Generally, the POS payment would be used for payment to merchants for goods while the subscriber-to-subscriber (STS) payment would be used to pay individuals or to pay bills.

Update Credit from Issuing Bank

Figure 13 shows an Update Credit from Issuing Bank transaction protocol, and particularly describes how subscribers acquire a credit note which is a preauthorized credit payment vehicle. A subscriber can carry up to one credit note for each credit account the subscriber owns. Note that every bank which allows a subscriber to receive updated credit notes is an issuing bank for those credit notes.

The process flow to set up an update credit transaction begins with a Setup Credit process between a money module A and a bank's Teller money module B, (step 1854), which is now described further with reference to Figure 14.

Setup Credit

The Setup Credit Withdrawal process begins when owner/holder of transaction money module A decides to make a credit update and thus signs onto his money module (step 1876). For instance, subscriber A may possess a credit note but wish to change (e.g., increase or decrease) the credit amount including decreasing the amount to zero, or may not currently possess a credit note but wish to acquire one. To Subscriber A prompts the owner/holder for a transaction, and A chooses to make a credit update of a particular amount from a particular bank and account (steps 1878 - 1880). In this implementation, the credit update amount specified by subscriber A is the total credit amount that subscriber A wishes to carry. Then, the Transaction money module A initiates a procedure for communicating with the bank that was selected by engaging the network via the Network Sign On procedure described hereinabove (step 1882).

After the steps of money module sign-on, transaction selection, and network sign-on are completed, A and B then establish a secure session (step 1884).

Then, Transaction money module A makes a credit request from the Teller money module B (step 1886), according to a Request Credit procedure more fully described with reference to Figure 15.

Request Credit

Referring now to Figure 15, a process for requesting credit will now be described. It should be noted that although the figure denotes the parties as "X" and "Y," in the process steps described below, they are applicable to any money module transacting with a Teller money module.

To begin, if there is a credit note for the selected account Note Directory X sends the amount of this credit note to To Teller X (step 1897). To Teller X determines the net difference between the total credit amount requested by subscriber A and the credit note amount and sends a credit update request to the Teller money module, requesting a certain net amount of credit to be authorized from a specific account. In it's transmission of the update credit request, the account number and the account profile will be transmitted from the requesting money module to the Teller money module along with the net credit amount (step 1898). This message is sent according to the Send Message protocol (step 1900), in which the message is encrypted using the described cryptographic techniques.

Once the credit withdrawal request and the account number and profile are transmitted to the Teller money module, a procedure to validate the account number is initiated (Step 1902). A flow diagram depicting how an account number is validated is shown in Figure 20, which is described separately below for clarity of exposition.

With the account information validated, To Bank Y verifies that there is sufficient credit to support the credit update request amount (Step 1904). Sufficient credit will prompt To Transaction Y to send an acknowledgement to X, which receives the acknowledgement via its To Teller application function (steps 1906-1912).

An insufficient amount of credit, however, will cause the subscriber to be prompted to enter a new amount for the credit update (steps 1914-1918, Figure 15B). Entry of a new amount for credit update by the subscriber results in the To Teller application sending the new credit amount to the To Bank application of the Teller money module to verify if there are sufficient funds to cover the latest requested amount (steps 1922 - 1924), returning to Step 1904 of Fig. 15A. If no new amount is requested by the subscriber, then the transaction is aborted (step 1926).

Referring back to Figure 14, upon return from the Request Credit Withdrawal process, To Teller A invokes a transfer of the total currency notes, transferred credit notes (i.e., credit notes received in previous transactions) and credit note for the account to the Teller money module (Step 1888). If there are no notes being held in the Transaction money module at the time the credit withdrawal request is made, the To Teller A application sends a message to the Teller money module that there are no notes present (steps 1892-1894). If there are notes being held in the Transaction money module, however, then the electronic notes are transferred from A to Teller B according to the Transfer Notes procedure described hereinabove with reference to Figure 8 (step 1896).

Returning to Figure 13, To Transaction B checks if any currency notes and transferred credit notes have been transferred (steps 1856 - 1858), and if any of these type of notes have indeed been transferred from Transaction money module A, accounting transactions are posted to reflect this situation by the To Bank application B (step 1860). Both in the case when no notes have been transferred from the money module and after the accounting transactions are posted in step 1860, a session is established between the Teller money module and the Money Generator module (step 1862). To Bank B updates the credit line by adding the amount of the credit note (if any) to the available line of credit to get the total available credit and deducting the requested credit amount from the total available credit. If no notes (including currency notes and credit notes) are to be created because the requested credit amount was zero and no currency notes were transferred, then the money modules will finalize the transaction according to the Commit procedure described hereinabove with reference to Figure 10 (steps 1865 - 1875).

If, however, any notes (credit or currency) are to be created, due to a nonzero credit amount request and/or transferred currency notes, then notes are requested by Teller B from Money Generator module according to the Request Notes procedure (steps 1865 - 1866). The requested notes in the Money Generator module are transferred to the Teller money module B using the Transfer Notes process outlined above (see Figure 8) for transferring electronic notes (step 1868). The notes are then transferred from the Teller money module B to the Transaction money module using the same Transfer Notes process (step 1870). Finally, to successfully complete the update credit procedure, the money modules will finalize the transaction according to the Commit procedure described hereinabove with reference to Figure 10. The Commit process is initiated first by the Transaction money module committing its transaction with the Teller money module B (step 1872). Then, the Commit process is executed between Teller money module B and the Money Generator module (step 1874). That completes the processing for one complete credit update from an Issuing Bank.

Money Issued Reconciliation

The process of note reconciliation checks for note counterfeiting and duplication, and is described with reference to Figure 16 and Figure 17. The Money Issued Reconciliation system, based on the information stored in the Money Issued Master file, constructs a note transfer tree which models the transfer history of the note.

Figure 16 schematically illustrates a hypothetical series of transactions among a Money Generator Module having an identifier number "1" (referred to as Money Generator 1), a teller money module having identifier "2" (referred to as teller module 2), and four transaction money modules having integer identifiers 3 through 6 (referred to as transaction modules 3-6), associated with a single note generated by Money Generator 1 at the date/time indicated by 1:00:00.

In accordance with the example of a transfer history shown in Figure 16, Figure 17 illustrates how the transfer of an electronic representation of currency produces a tree-like structure of electronic representations of currency derived from the initial note produced by a money generator module. As individual transfers (part of a tree branch) of the note are deposited or return to the banking system according to note updating, the note transfer tree in figure 17 is built by the Money Issued Reconciliation system. In this example, money generator 1 (module identifiers are contained in digitally signed certificates) produces an electronic representation of currency 2300 having a body group of data fields and a transfer group of data fields, which are shown schematically in part for purposes of clarity of exposition. In addition, the signatures and certificates group of data fields is not shown for convenience.

The body group of data fields includes a note identifier (e.g., "N12"), a money generator module identifier (e.g., "MG1"), an issuing bank identifier (e.g.,"Bank Name"), a date-of-issue (e.g., 1:00:00), a date-of-expiration (e.g., 12:00:00), a note amount and a monetary unit identifier (e.g., $50). Other body group data fields such as type of note are not shown for convenience. The various date fields in the electronic notes are shown for illustrative purposes as being in the form day:hr:min. Other time monitoring forms (e.g., including seconds) are, of course, possible.

The transfer group of data fields includes a transfer record having a transferee identification number, a date-of-transfer, and a transfer amount. The transfer group also preferably includes a sequence number which is incremented by the note directory of the transferor after each transfer. Normally, the date/time and transfer id of the transfer should be sufficient to uniquely identify a transfer. However, it might be possible that a transfer, transferee id, date/time, and amount could duplicate if there is a time adjustment between transfers and the same amount is transferred to the same module. Thus, to avert this potential problem, a sequence number (seq) is preferably added to the transfer record and the note directory to uniquely identify the transfer. The sequence number will be incremented by the note directory after every transfer. If a sequence number is reset, then this would be detected as a duplicate.

Thus, when electronic representation of currency 2300 is transferred to teller module 2, transfer record 2302 is appended to the transfer group, and includes the transferee identification number (e.g., "2"), the date-of-transfer (e.g., 1:00:00), the transfer amount (e.g., $50), and the sequence number (e.g., "1"). For illustrative convenience, the note transfers represented in Figure 17 only show the newly appended transfer record portion of the transferred note. Also for convenience, the transfer group data field indicating total number of transfers is not shown.

The electronic representation of currency 2300 from money generator 1 is stored in a teller module 2. As part of the withdrawal of $50 by transaction module 3, teller module 2 forms an electronic representation of currency by appending transfer record 2304 to a copy of the data fields in the electronic representation of currency 2302 augmented by transfer record 2302. This note is then stored in transaction module 3 upon completion of the withdrawal. It may be understood that each node of the note transfer tree shows the newly appended transfer record portion of the transferred note.

As represented by the note transfer tree, at 1:00:05, transaction module TR3 pays $10 by transfer record 2306 to transaction module 4. At 1:01:00, transaction module 3 pays $10 by transfer record 2308 to transaction module 5. At 3:08:01, transaction module 3 pays $25 by transfer record 2310 to transaction module 5. At 4:11:08, transaction module 3 transfers $5 by transfer record 2312 to transaction module 6.

At 2:00:01, transaction module 4 transfers $5 by transfer record 2314 to transaction module 6. At 2:01:07, transaction module 4 transfers an additional $5 by transfer record 2315 to transaction module 6, which in turn, at 3:07:05 transfers $5 by transfer record 2321 to transaction module 3.

At 2:00:06, transaction module 5 transfers the entire $10 note to transaction module 3 by transfer record 2316. From the $25 note received at 3:08:01 by transaction module 5 from transaction module 3, at 3:09:12 transaction module 5 pays transaction module 6 $20 by transfer record 2318, and deposits the remaining $5 to teller module 2 at 4:12:05 by transaction record 2320.

At 4:10:00, transaction module 6 transfers $10 to transaction module 5 according to transfer record 2322, and at 5:00:06 transfers the remaining $10 to transaction module 3 by transfer record 2324. In accordance with an embodiment of the present invention, it is understood that upon deposit of money from a transaction module to a bank, all notes (including credit notes) in the transaction module are sent to the banking system and are updated. Therefore, substantially simultaneous with the above described deposit from transaction module 5 to teller module 2 represented by transfer record 2320, an additional and concurrent transfer represented by transfer record 2326 occurs automatically. Then, a new note having a value of $5 (assuming transaction module 3 had no credit notes) will be generated by money module 1 and transferred to transaction module 3 via teller module 2, with the appropriate transfer records appended (not shown). Accordingly, it may be appreciated that updating all notes in a transaction money module upon a transaction (e.g., deposit or withdrawal) between the transaction module and a teller module facilitates the note reconciliation process by providing an additional means for returning notes to the banking system.

At 5:00:10 transaction module 3 deposits $10 to teller module 2 by transaction record 2328. As described above for the deposit by transaction module 5, concurrent with the deposit by transaction module 3 represented by transaction record 2328, additional and concurrent transfers (not shown) to the banking system of all notes possessed by transaction module 3, including those represented by transfer record 2316 and transfer record 2321, occurs. Then, the banking system returns to transaction module 3 a note having a value equal to the total notes sent to the banking system for updating (e.g., $15).

Thus, at this point in time, only transaction module 6 possesses transferable vestiges of original note 2300, as represented by transfer notes 2312 and 2314. If transaction module 6 transacts (e.g., deposit or withdrawal) with a teller module before transferring these notes to other transaction money modules, then there will be no transferable notes in circulation that relate to original note 2300; all notes derived from transfers of original note 2300 will have been returned to the banking system, permitting complete construction of the note transfer tree shown in Figure 17. The date-of-expiration effectively facilitates note reconciliation by limiting the time that a note may be transferred.

From the note transfer tree, it may be understood that if the note were counterfeit then there would be no note body to match the first piece deposited. If a transfer were duplicated then the transfers below would sum to a value greater than the superordinate transfer amount. For example, if transaction module 6 transferred $20 to transaction module 3 at 5:00:06, instead of $10 (i.e., transfer record 2324), then the transfers below transfer record 2318 (i.e., SEQ1, 3:09:12, TR6, $20) would sum to $30 denoting that transaction module 6 had duplicated the transfer.

Link Money Module to Bank Account(s) for Bank Access

Figure 18 shows the protocol for linking a money module to bank accounts for bank access. The process begins when a subscriber identifies him/herself to a customer service representative (CSR) and requests that the CSR link the subscriber's accounts to a money module (step 1928). The CSR enters a request into CSM host A (CSMHA) to link accounts for the identified subscriber, and CSMHA accesses the identified subscriber's account information from the bank systems (steps 1930 - 1934). Then, the subscriber and the CSR verify the account information, and the subscriber selects which accounts to link to the money module (step 1936).

After the subscriber requests that his/her money module B link bank accounts and the CSR requests, via CSMHA, that CSMA link bank accounts, a secure session is established between the subscriber's money module B and the CSMA (steps 1938 - 1946). Then, in response to a request from To Host A of CSMA, HCSMA sends the account information to CSMA, which receives the account information and constructs an account profile therefrom (steps 1948 - 1952). Public Key A then signs the account profile, and Create Account Profile constructs a message from the account profile and signature and sends this message to money module B (steps 1954 - 1958). Maintain Security B receives the message, and Public Key B checks the digital signature on the message (steps 1958 - 1962). If the signature is invalid, then the session is aborted (step 1966).

If the signature is valid, then To Subscriber B sends the profile to the host in order for the customer to verify the account profile. If the customer does not affirm the account profile, then the transaction is aborted. If the customer affirms the account profile, Maintain Security B adds the CSM certificate to the account profile (step 1968).

To Teller B then checks to determine whether an account profile is already stored for the bank associated with the recently created ("new") account profile. If an account profile for the bank already exists in the To Teller B application, then it is replaced by To Teller B with the new account profile; otherwise, To Teller B adds the new account profile. (Steps 1970 - 1974).

Revalidate Money Module Link to Bank Accounts

Figure 19 shows the protocol for a subscriber to revalidate the subscriber's money module link to bank accounts. The process begins when the subscriber signs on to his/her money module, and in response to a prompt for a transaction generated by To Subscriber A, the subscriber chooses to revalidate a bank account link for a bank associated with a customer service module (CSM) B (steps 1978 - 1982). The money module invokes and executes the network sign-on protocol described with reference to Figure 6, hereinabove, and a secure session is established between money module A and CSMB (step 1986). To Teller A then sends the account profile for the bank accounts to CSMB (steps 1988 - 1990). Create Account Profile B receives the message, and Maintain Security B validates the CSM certificate and the signature of the account profile (steps 1992 - 1995). If the certificate or signature is invalid, then the CSM aborts the transaction (step 2000). If the certificate is valid, To Host B sends the list of accounts from the account profile to the CSM host (CSMH), which checks with the on-line banking system to determine whether each account is currently active (steps 1996 - 2001). If any of the accounts has expired, CSMH signals an abort message to CSM (step 2010), which then aborts the transaction according to the Abort process (step 2000).

If all the accounts are active, then CSMH signals a revalidate instruction to CSM, and Create Account Profile B receives the message and constructs an account profile from the account information (steps 2002 - 2004). Public Key B then signs the account profile, and Create Account Profile B constructs a message from the account profile and signature and sends this message to money module A (steps 2006 - 2010). Public Key A receives the message and verifies the digital signature (step 2012). If the signature is invalid, money module A aborts the transaction (step 2018); if it is valid, the profile signature and the CSM certificate is added to the account profile (step 2014), and money module A commits the transaction (step 2016).

Validate Account Number

In accordance with an embodiment of the invention employing a customer services module (CSM), which is described hereinabove, a flow diagram depicting how an account number is validated is shown in Figure 20.

In this process, Maintain Security Y receives the account number and account profile, including the CSM certificate, and verifies the CSM certificate (step 2020). An invalid certificate causes the transaction between the two money modules to be aborted (step 2028).

If the certificate is valid, Maintain Security Y conveys the account profile to Public Key Y to verify the CSM signature (steps 2022). An invalid signature causes Maintain Security Y to inform the Session Manager that the account profile is invalid (Step 2026), and the transaction between the two modules is aborted (step 2028).

If the signature test confirms a valid signature, the procedure advances to To Bank Y which sends the account number it has received to the bank's on-line computer system (step 2024). An inactive account will cause Maintain Security Y to inform the Session Manager of the inactive account (step 2030) and have the transaction aborted (step 2028); an active account will cause the Validate Account Number process to return to the next step in the operational flow that invoked the Validate Account Number process.

As can be seen, the Validate Account Number process is simplified for the Teller money module as compared to an embodiment of the present invention that does not include a CSM.

Claim Lost Money

As has been discussed, electronic money could be lost for any of several reasons, including: (1) the money module is damaged and no longer functions; (2) the money module is lost or stolen; or (3) the commit fails. It is important to the utility of a monetary system that the subscriber to the system feel confident that their money is safe. Thus it is important that the transferee be able to reclaim money lost through system failure. Being able to replace money when the money module is damaged would improve confidence since electronic failure is more likely than the loss of a paper by physical damage. Replacing money because of mishaps due to a lost or stolen money module is more problematic. Allowing such claims could invite widespread claims on the system, since subscribers would not take precautions against loss.

In any case, embodied in the present invention are methods that allow money lost in either of these cases to be replaced. In the first two cases (i.e., cases (1) and (2) above), the subscriber would have to periodically create a lost note(s) claim (see figure 21) which could be stored outside the money module. Upon failure, the claim could be submitted with the subscriber's identity to an issuing bank (see figure 22). The claim would contain the last known state of the subscriber's money module. A description of the claimed notes could be validated and distributed to the issuing banks. The issuing banks could replace the money after a period of time if the claimed notes have not been deposited.

In the case wherein the commit fails (i.e., case (3)), the claim is created interactively between the money module and a customer service module, if the money module is still functioning (see figure 22). This claim, as in cases (1) and (2), is passed to the Money Issued Investigation System, which monitors the claim against money deposited. The issuing banks can feel confident in replacing lost money because they can monitor the incoming money against fraudulent applications and they have the applicant's identity.

These methods are more fully described with reference to Figure 21 and Figure 22.

Create Lost Note(s) Claim

Referring now to Figure 21A, there is shown a Create Lost Note(s) Claim process which may be practiced in accordance with an embodiment of the present invention. The process begins when a subscriber signs on to a money module, To Subscriber A prompts the subscriber for a transaction, and the subscriber selects a Create Lost Note(s) Claim option (steps 2032 - 2036).

Then, several steps occur which consolidate all notes and unclaimed failed commits (steps 2038 - 2042). Specifically, Note Directory A creates a unique claim sequence number (which is used for identifying a claim) and conveys a copy of the Note Directory with the sequence number to the Packet Manager. Notes A conveys a copy of all notes with signatures and certificates to Packet Manager. Then, Tran Log A sends all unclaimed failed to Commit transactions that were logged during the Abort process to the Packet Manager.

Next, Public Key A uses a money module private key to sign the claim sequence number, note directory, notes, and failed Commits that were sent to the Packet Manager, and Packet Manager A appends the signature to the assembled data, resulting in an assembled data packet (steps 2044 - 2046), which is then encrypted by Public Key A (step 2048). To the encrypted claim, Public Key A then appends a claim description consisting of the claim sequence number, the total value amount of the claim, and the certificate of money module A (step 2050). To Subscriber A sends this claim to the money module host, which receives and stores the claim--on a medium physically independent of the money--for future use (steps 2052 - 2054).

It is understood, therefore, that the Claim Lost Notes process provides a method for generating and protecting potential money claims which are not destroyed by damage to, or failure of, a money module.

Claim Lost Notes

Figure 22 shows the protocol for claiming lost notes, which begins when a subscriber requests to make a claim for lost notes to a customer service representative (CSR), the subscriber's identity being revealed to the CSR (step 2056). CSR then conveys the subscriber's identity to customer service module host (CSMH) A, and checks whether the claim is due to a failed commit or due to a loss or destruction of the money module (steps 2058 - 2060). If a failed commit is the basis for the claim (and the subscriber's money module has not been lost or damaged), then upon the subscriber's selection to make a claim for lost notes from the money module and the CSR's selection to receive a claim for lost notes from a money module, Session Manager B of the subscriber's money module and Session Manager A of a customer service module (CSM) associated with the customer service representative (via a customer service module host (HCSM)) are respectively invoked to establish a secure session between CSM A and money module B (steps 2062 - 2070).

Once a secure session is established, To Host A of the CSM requests the identity of the subscriber, and HCSM A responds to this request by sending a message containing the subscriber identity to CSM (2072 - 2074). Claim Lost Notes A receives this message and sends to money module B a message indicating that the claim should be sent (steps 2076 - 2078).

Tran Log B receives this message and retrieves failed commit records that have not been claimed (step 2080). If no failed commit records are located, then the transaction is aborted (step 2083). Otherwise, To Subscriber B sends information (e.g., date, time, and amount) from the retrieved failed commit records to the subscriber for review (step 2082). From this information, the subscriber selects the transactions to be claimed (step 2084). For instance, the subscriber would not select failed commit transactions that have been resolved separately. Then, for each failed commit transaction selected by the subscriber, Tran Log B constructs a message containing log information for these transactions, and sends this message to CSM (steps 2086 - 2088).

Claim Lost Notes A receives this message and sends to B an acknowledgement including a claim identifier (e.g., a confirmation number) for future reference to the claim (step 2092). Tran Log B receives the acknowledgement and marks each selected failed commit transaction record with the date/time of the claim and with the claim identifier provided by the CSM (step 2094). Then, B commits to the transaction (step 2096).

After the commit procedure is completed, from the received message containing failed commit transaction information Claim Lost Notes A constructs a claim to be sent to the Money Issued Investigation System (MIIS) which preferably forms part of the Transaction Investigation System. To Host A sends this claim to HCSMA, which receives the claim and further sends the claim to the MIIS. (Steps 2098 - 2102).

Returning now to step 2060 of Figure 22, if the claim were not due to a failed commit (e.g., due to a lost or damaged money module), then the subscriber may choose to make a claim for lost notes from a host B that has access to any of the subscriber's claims that were generated and off-loaded from the money module computing environment (e.g., onto a FLASH memory) in accordance with the Create Lost Note(s) Claim process described hereinabove with reference to Figure 21. CSR separately chooses to initiate a process to receive a claim for lost notes from a money module host and the hosts establish a communication link according to any known technique (steps 2104 - 2108).

Then, the subscriber host B sends the claim, including the money module certificate, to CSMHA which sends the claim to CSMA (steps 2108 - 2112). Claim Lost Notes A receives the claim and sends the claim to Public Key A which validates the money module certificate (2114 - 2116). If the money module certificate is invalid, then To Host A sends CSMHA a message indicating that the claim has been rejected, and CSMHA routes this message to the subscriber host B, and the Claim Lost Notes process is terminated (steps 2120 - 2121).

If the money module certificate is valid, Public Key A decrypts the claim and verifies all note certificates and signatures within the claim (step 2118). If any of the note certificates and signatures are not valid then the transaction is terminated (steps 2119 - 2121). If the note certificates and signatures are valid, then Claim Lost Notes A, verifies the consistency of transfer amounts to ensure that the value of the money conveyed to a transferee is not greater than that received by the transferor for the transfer history of each note (step 2122 - 2123). If there is an inconsistency then the transaction will be terminated (steps 2120 - 2121).

If, however, the transfer amounts are consistent for all notes, then Claim Lost Notes A constructs a claim to be sent to the MIIS, and also generates a claim identifier to be associated with the claim. To Host A then sends the constructed claim and claim identifier to CSMHA, which receives the claim and claim identifier and appropriately routes the claim identifier to subscriber host B and sends the claim to the MIIS. (Steps 2124 - 2128), thus completing the Claim Lost Notes process.

Although the above description provides many specificities, these enabling details should not be construed as limiting the scope of the invention, and it will be readily understood by those persons skilled in the art that the present invention is susceptible to many modifications, adaptations, and equivalent implementations without departing from this scope and without diminishing its attendant advantages. It is therefore intended that the present invention is not limited to the disclosed embodiments but should be defined in accordance with the claims which follow.

THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS:-

1. A method for a first subscriber to exchange an electronic representation of a first foreign currency stored in a first transaction module for an electronic representation of a second foreign currency stored in a second transaction module, comprising the steps of:

(a) establishing a cryptographically secure session between said first transaction module and said second transaction module;

(b) said subscriber selecting, by way of said first transaction module, a first amount of said first foreign currency to be sold;

(c) checking if said first transaction module has sufficient funds;

(d) said first transaction module sending said first amount to said second transaction module, via said cryptographically secure session;

(e) said second transaction module prompting its owner to select an exchange rate or a second amount of said second currency;

(f) checking if said second transaction module has sufficient funds;

(g) said second transaction module sending a message indicative of said second amount or said exchange rate to said first transaction module, via said cryptographically secure session;

(h) said first subscriber confirming said second amount or said exchange rate;

(i) said first transaction module sending said electronic representation of first foreign currency to said second transaction module, in said first amount, via said cryptographically secure session;

(j) said second transaction module sending said electronic representation of said second foreign currency to said first transaction module, in said second amount, via said cryptographically secure session; and

(k) committing said first transaction module to transferring said first foreign currency to said second transaction module and said second transaction module to receiving said first foreign currency from said first transaction module, and committing said second transaction module to transferring said second foreign currency to said first transaction module and said first transaction module to receiving said second foreign currency from said second transaction module, in an order which is not predictable.

2. The method according to claim 1, wherein step (k) comprises the steps of:

(a) said first transaction module and said second transaction module sharing a common random binary value, having either a first value or a second value;

(b) said first transaction module conditionally logging either said transferring of said first foreign currency to said second transaction module or said receiving of said second foreign currency from said second transaction module, where said common random binary value has said first value or said second value, respectively, such that rolling back may occur;

(c) said first transaction money module sending a message to said second transaction module indicating that said first transaction module has completed conditionally logging a transfer;

(d) said second transaction module, in response to said message, conditionally logging either said receiving of said first foreign currency from said first transaction module or said transferring of said second foreign currency to said first transaction module, where said common random binary value equals said first value or said second value, respectively, such that rolling back may occur;

(e) said second transaction money module, where said common random number value equals the first value, sending a start-commit message to said first transaction module;

(f) said first transaction module, in response to said start-commit message, setting said conditionally logging to unconditional logging, and initiating a commit protocol wherein said first transaction module commits to its outstanding transfer so that said first transaction module can no longer abort said transfer by rolling-back its state, and wherein said second money module commits to said transferring second foreign currency and to said receiving first foreign currency by logging said transferring and receiving so that said second money module can no longer abort said exchange of foreign currency by rolling-back its state; and

(g) said second transaction module, where said common random number value equals the second value, setting said conditionally logging to unconditional logging and initiating a commit protocol wherein said second transaction module commits to its outstanding transfer so that said second transaction module can no longer abort said transfer by rolling-back its state, and wherein said first money module commits to said transferring first foreign currency and receiving second foreign currency so that said first money module can no longer abort said exchange of foreign currency by rolling-back its state.

3. The method according to claim 2, wherein said common random number is generated as a session key during said step of establishing a cryptographically secure session.

4. The method according to claim 1, wherein said commit protocol initiated by a given money module for any outstanding money transfer with respect to another money module, comprises the steps of:

said given module sending a ready-to-commit message to said another money module;

said another money module sending an acknowledgement to said given money module in response to said ready-to-commit message;

said given money module committing by logging said any outstanding money transfer so that said given money module can no longer abort said any outstanding money transfer by rolling-back its state; and

said another money module committing by logging said any outstanding money transfer so that said another money module can no longer abort said any outstanding money transfer by rolling-back its state.

5. The method according to claim 1, wherein in step (b) said subscriber selects said first amount of said first foreign currency to be sold by type of note.

6. A method of exchanging electronic representations of foreign currencies for other foreign currencies in an electronic monetary system substantially as hereinbefore described with reference to the information disclosed in the accompanying drawings.

DATED this 30th Day of March, 2000

CITIBANK, N.A.

Attorney: JOHN B. REDFERN
Fellow Institute of Patent Attorneys of Australia
of BALDWIN SHELSTON WATERS

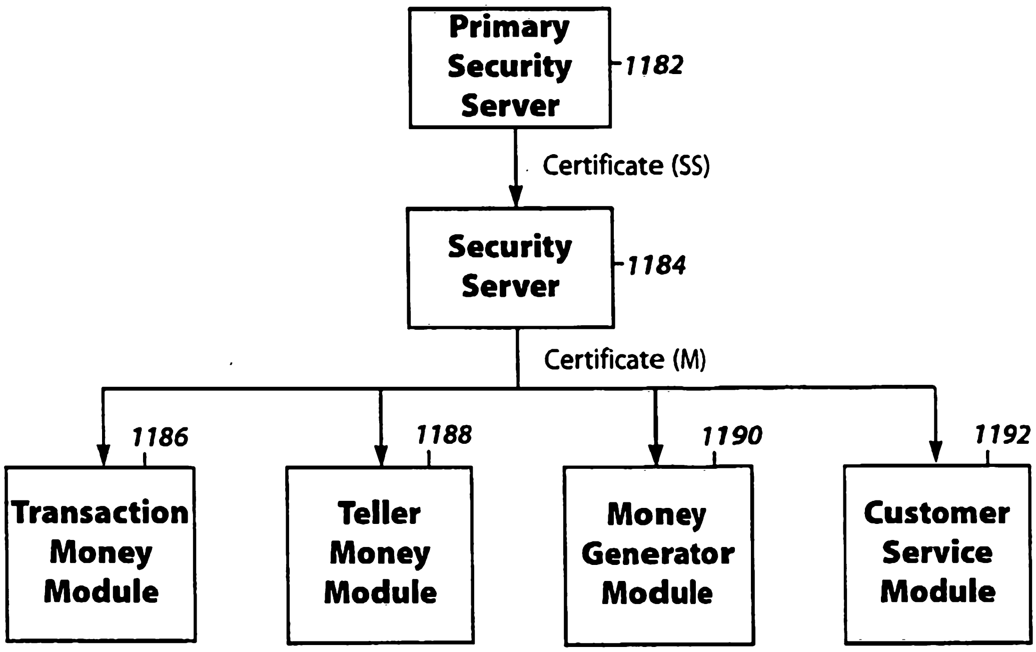


Figure 1A

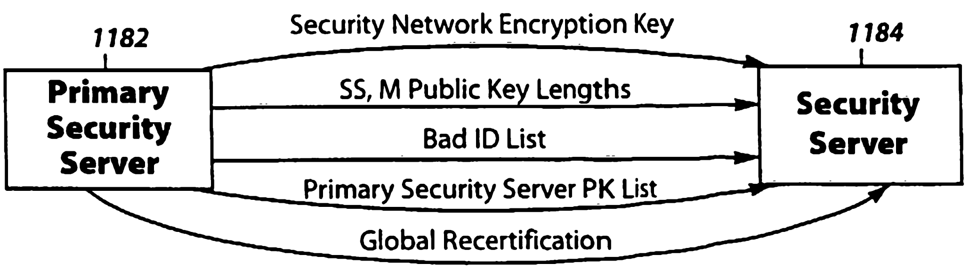
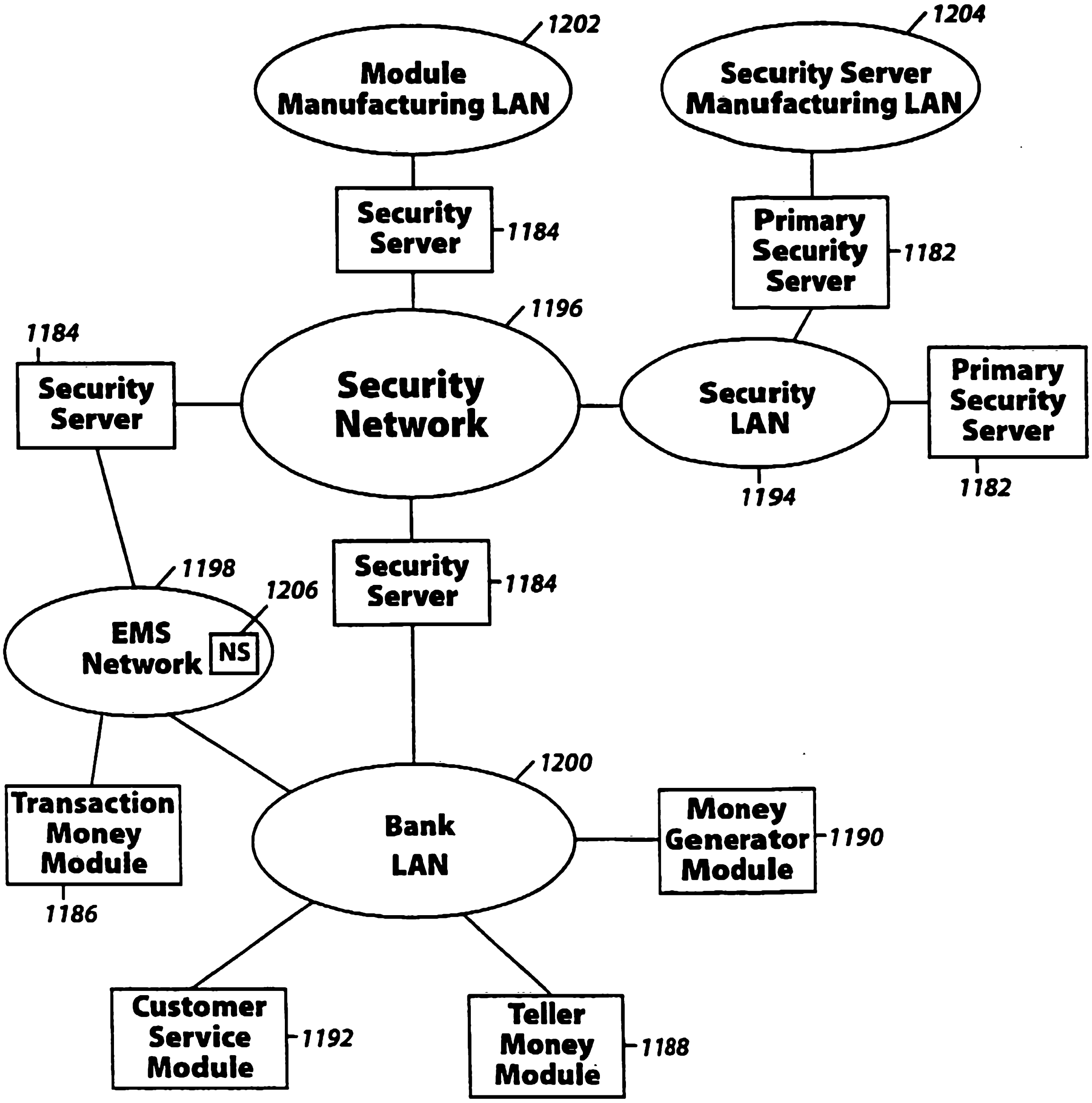


Figure 1B



1202
Module Manufacturing LAN
1204
Security Server Manufacturing LAN
Security Server
1184
Primary Security Server
1182
1196
1184
Security Server
Security Network
Security LAN
Primary Security Server
1194
1182
1198
1206
Security Server
1184
EMS Network
NS
1200
Transaction Money Module
1186
Bank LAN
Money Generator Module
1190
Customer Service Module
1192
Teller Money Module
1188

Figure 2

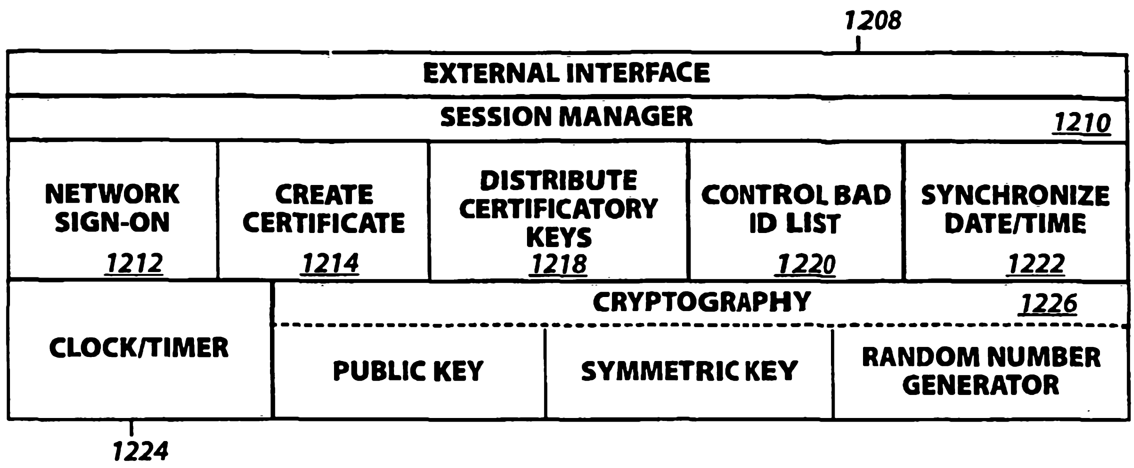


Figure 3A

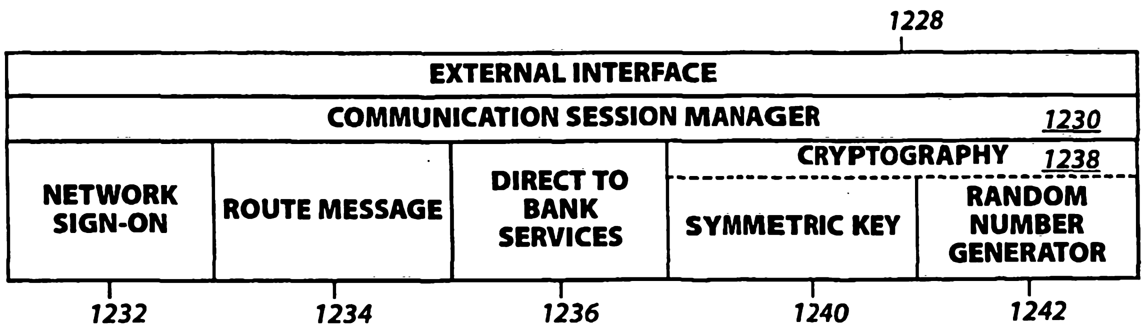


Figure 3B

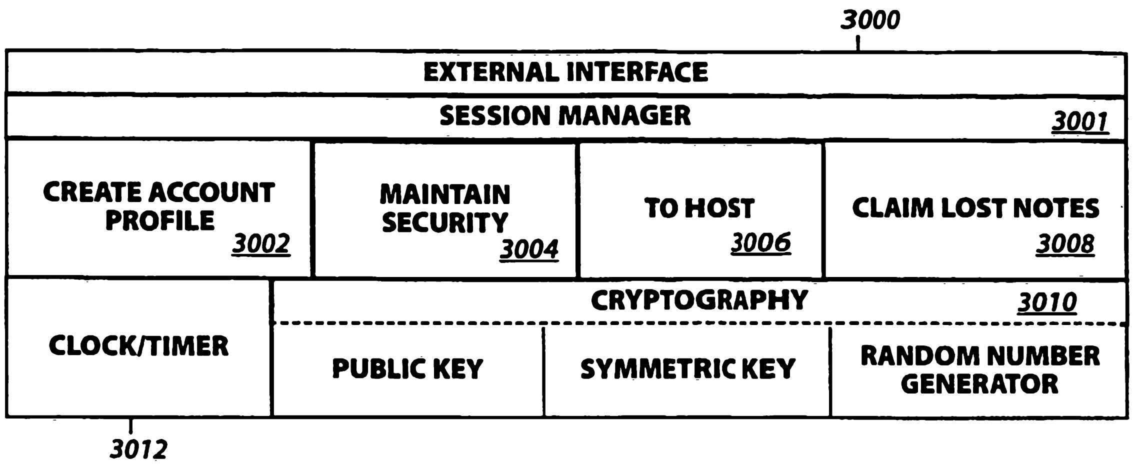


Figure 4A

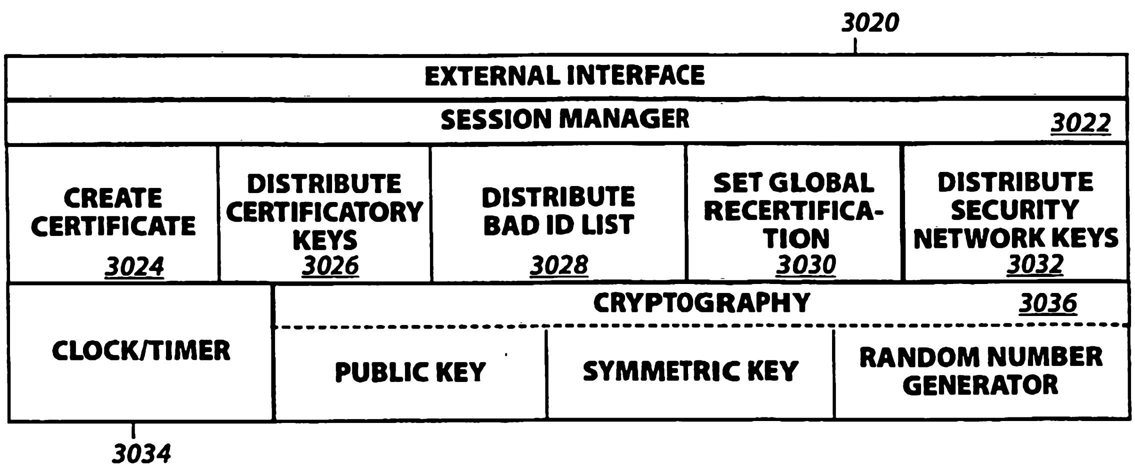


Figure 4B

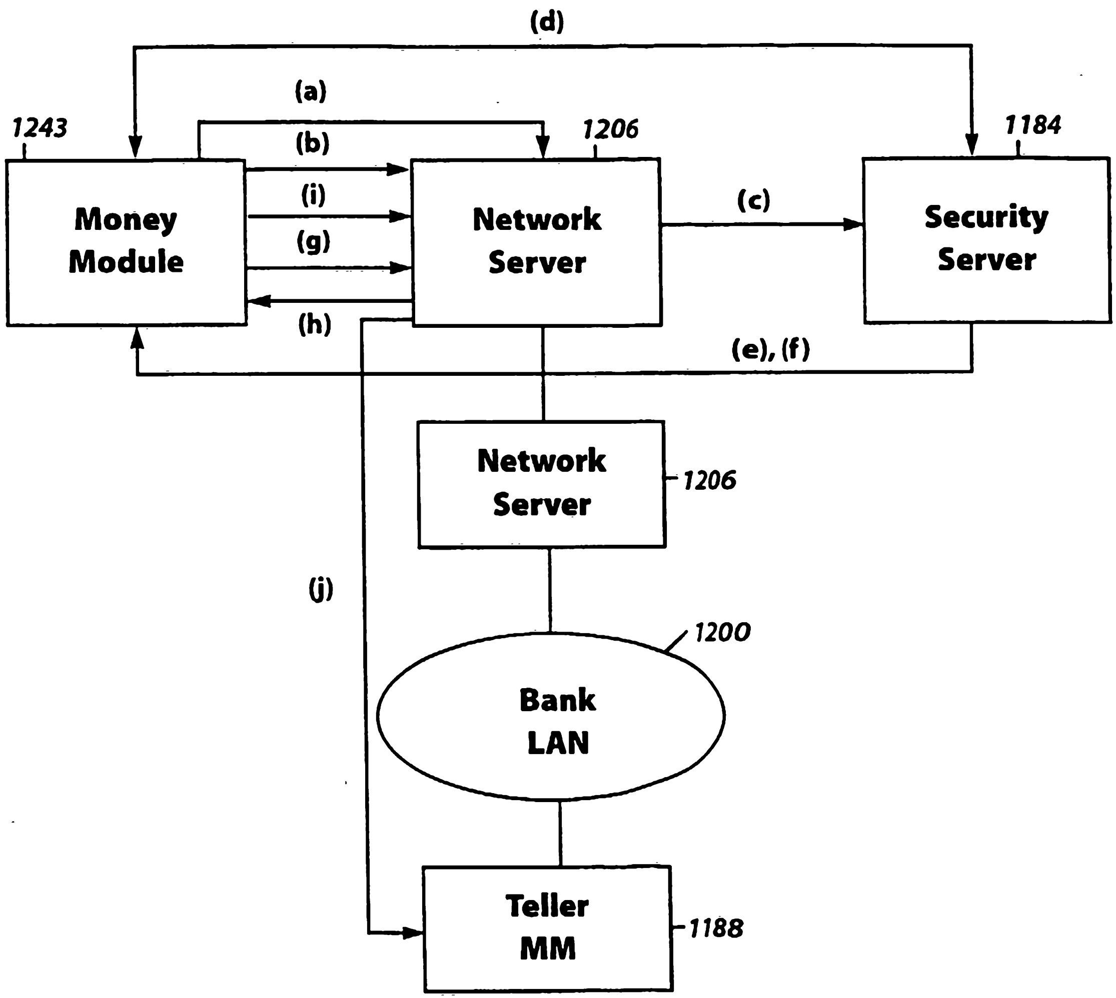

(d)
(a)
1243
(b)
1206
1184
Money Module
(i)
Network Server
(c)
Security Server
(g)
(h)
(e), (f)
Network Server
1206
(j)
1200
Bank LAN
Teller MM
1188

Figure 5

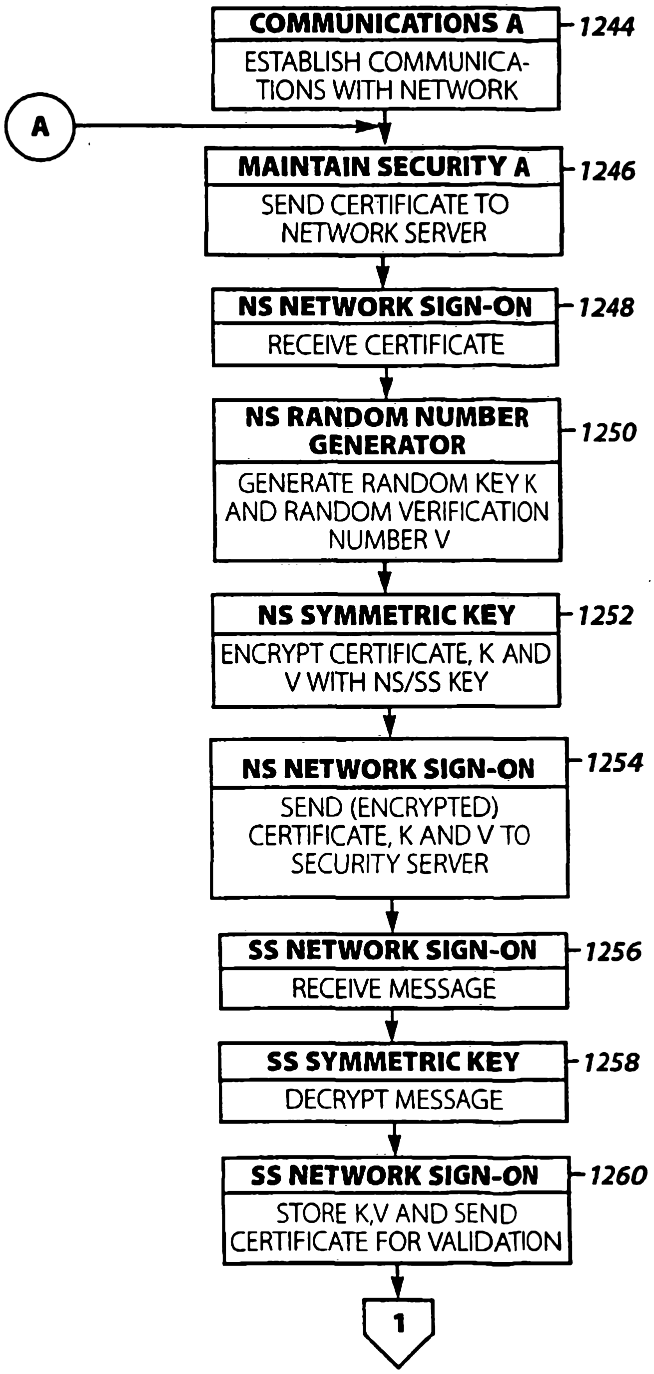

COMMUNICATIONS A
ESTABLISH COMMUNICATIONS WITH NETWORK
1244
A
MAINTAIN SECURITY A
SEND CERTIFICATE TO NETWORK SERVER
1246
NS NETWORK SIGN-ON
RECEIVE CERTIFICATE
1248
NS RANDOM NUMBER GENERATOR
GENERATE RANDOM KEY K AND RANDOM VERIFICATION NUMBER V
1250
NS SYMMETRIC KEY
ENCRYPT CERTIFICATE, K AND V WITH NS/SS KEY
1252
NS NETWORK SIGN-ON
SEND (ENCRYPTED) CERTIFICATE, K AND V TO SECURITY SERVER
1254
SS NETWORK SIGN-ON
RECEIVE MESSAGE
1256
SS SYMMETRIC KEY
DECRYPT MESSAGE
1258
SS NETWORK SIGN-ON
STORE K,V AND SEND CERTIFICATE FOR VALIDATION
1260
1

Figure 6A

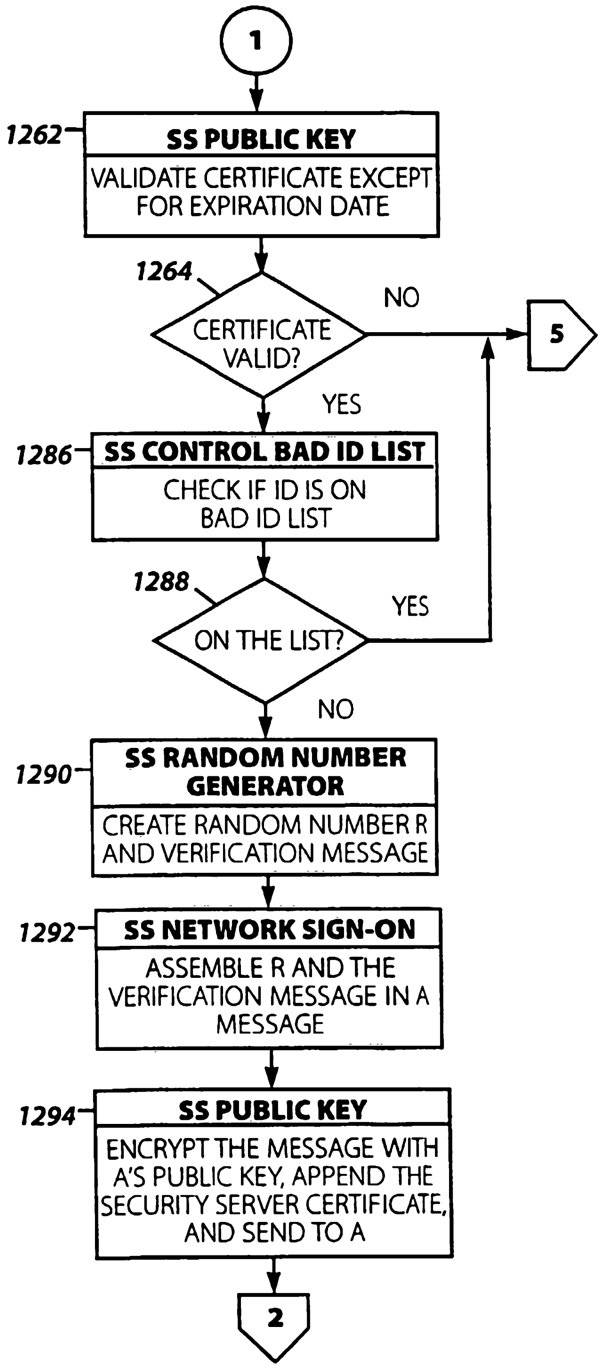

1
1262
SS PUBLIC KEY
VALIDATE CERTIFICATE EXCEPT FOR EXPIRATION DATE
1264
CERTIFICATE VALID?
NO
5
YES
1286
SS CONTROL BAD ID LIST
CHECK IF ID IS ON BAD ID LIST
1288
ON THE LIST?
YES
NO
1290
SS RANDOM NUMBER GENERATOR
CREATE RANDOM NUMBER R AND VERIFICATION MESSAGE
1292
SS NETWORK SIGN-ON
ASSEMBLE R AND THE VERIFICATION MESSAGE IN A MESSAGE
1294
SS PUBLIC KEY
ENCRYPT THE MESSAGE WITH A'S PUBLIC KEY, APPEND THE SECURITY SERVER CERTIFICATE, AND SEND TO A
2

Figure 6B

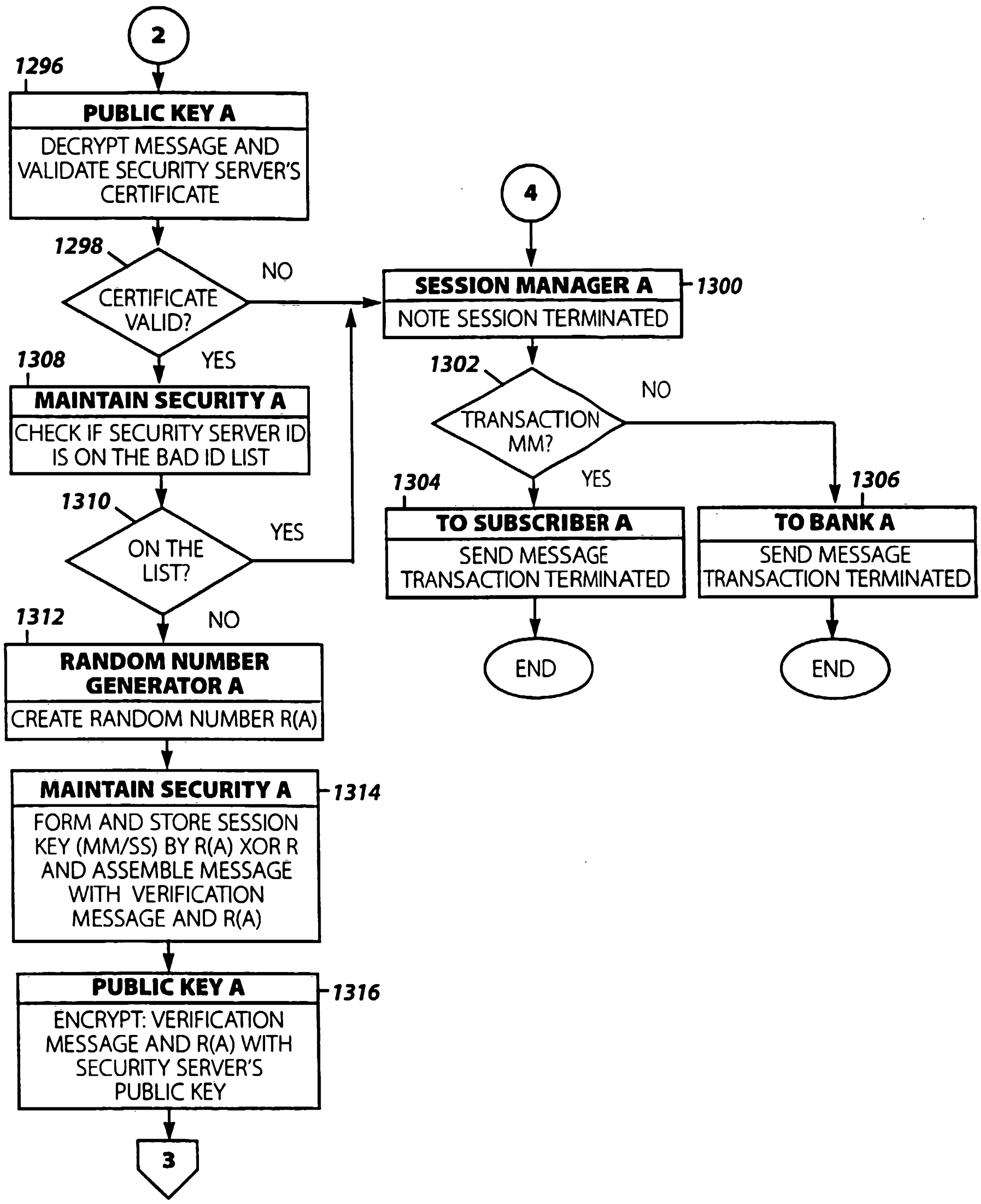

2
1296
PUBLIC KEY A
DECRYPT MESSAGE AND VALIDATE SECURITY SERVER'S CERTIFICATE
1298
CERTIFICATE VALID?
NO
YES
1308
MAINTAIN SECURITY A
CHECK IF SECURITY SERVER ID IS ON THE BAD ID LIST
1310
ON THE LIST?
YES
NO
1312
RANDOM NUMBER GENERATOR A
CREATE RANDOM NUMBER R(A)
MAINTAIN SECURITY A
1314
FORM AND STORE SESSION KEY (MM/SS) BY R(A) XOR R AND ASSEMBLE MESSAGE WITH VERIFICATION MESSAGE AND R(A)
PUBLIC KEY A
1316
ENCRYPT: VERIFICATION MESSAGE AND R(A) WITH SECURITY SERVER'S PUBLIC KEY
3
4
SESSION MANAGER A
1300
NOTE SESSION TERMINATED
1302
TRANSACTION MM?
NO
YES
1304
TO SUBSCRIBER A
SEND MESSAGE TRANSACTION TERMINATED
1306
TO BANK A
SEND MESSAGE TRANSACTION TERMINATED
END
END

Figure 6C

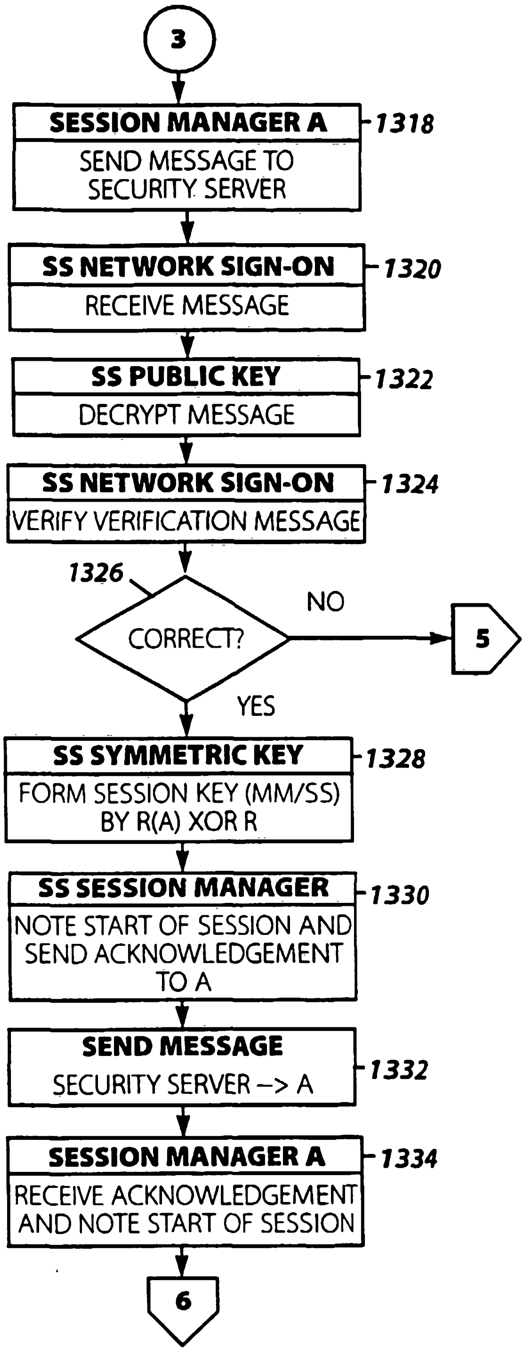

3
SESSION MANAGER A
SEND MESSAGE TO SECURITY SERVER
1318
SS NETWORK SIGN-ON
RECEIVE MESSAGE
1320
SS PUBLIC KEY
DECRYPT MESSAGE
1322
SS NETWORK SIGN-ON
VERIFY VERIFICATION MESSAGE
1324
1326
CORRECT?
NO
5
YES
SS SYMMETRIC KEY
FORM SESSION KEY (MM/SS) BY R(A) XOR R
1328
SS SESSION MANAGER
NOTE START OF SESSION AND SEND ACKNOWLEDGEMENT TO A
1330
SEND MESSAGE
SECURITY SERVER -> A
1332
SESSION MANAGER A
RECEIVE ACKNOWLEDGEMENT AND NOTE START OF SESSION
1334
6

Figure 6D

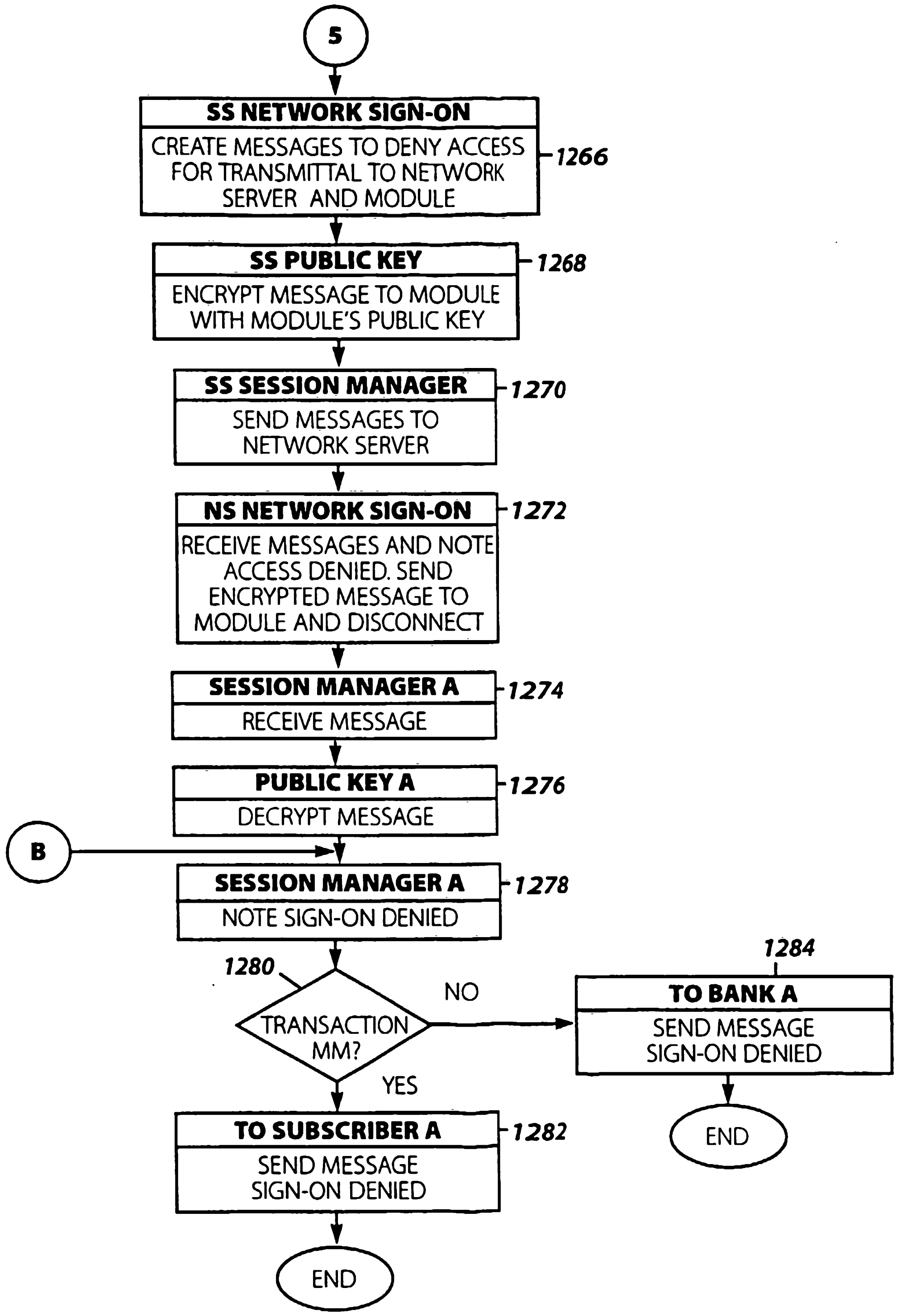

5
SS NETWORK SIGN-ON
CREATE MESSAGES TO DENY ACCESS FOR TRANSMITTAL TO NETWORK SERVER AND MODULE
1266
SS PUBLIC KEY
ENCRYPT MESSAGE TO MODULE WITH MODULE'S PUBLIC KEY
1268
SS SESSION MANAGER
SEND MESSAGES TO NETWORK SERVER
1270
NS NETWORK SIGN-ON
RECEIVE MESSAGES AND NOTE ACCESS DENIED. SEND ENCRYPTED MESSAGE TO MODULE AND DISCONNECT
1272
SESSION MANAGER A
RECEIVE MESSAGE
1274
PUBLIC KEY A
DECRYPT MESSAGE
1276
B
SESSION MANAGER A
NOTE SIGN-ON DENIED
1278
1280
TRANSACTION MM?
NO
YES
1284
TO BANK A
SEND MESSAGE SIGN-ON DENIED
END
TO SUBSCRIBER A
SEND MESSAGE SIGN-ON DENIED
1282
END

Figure 6E

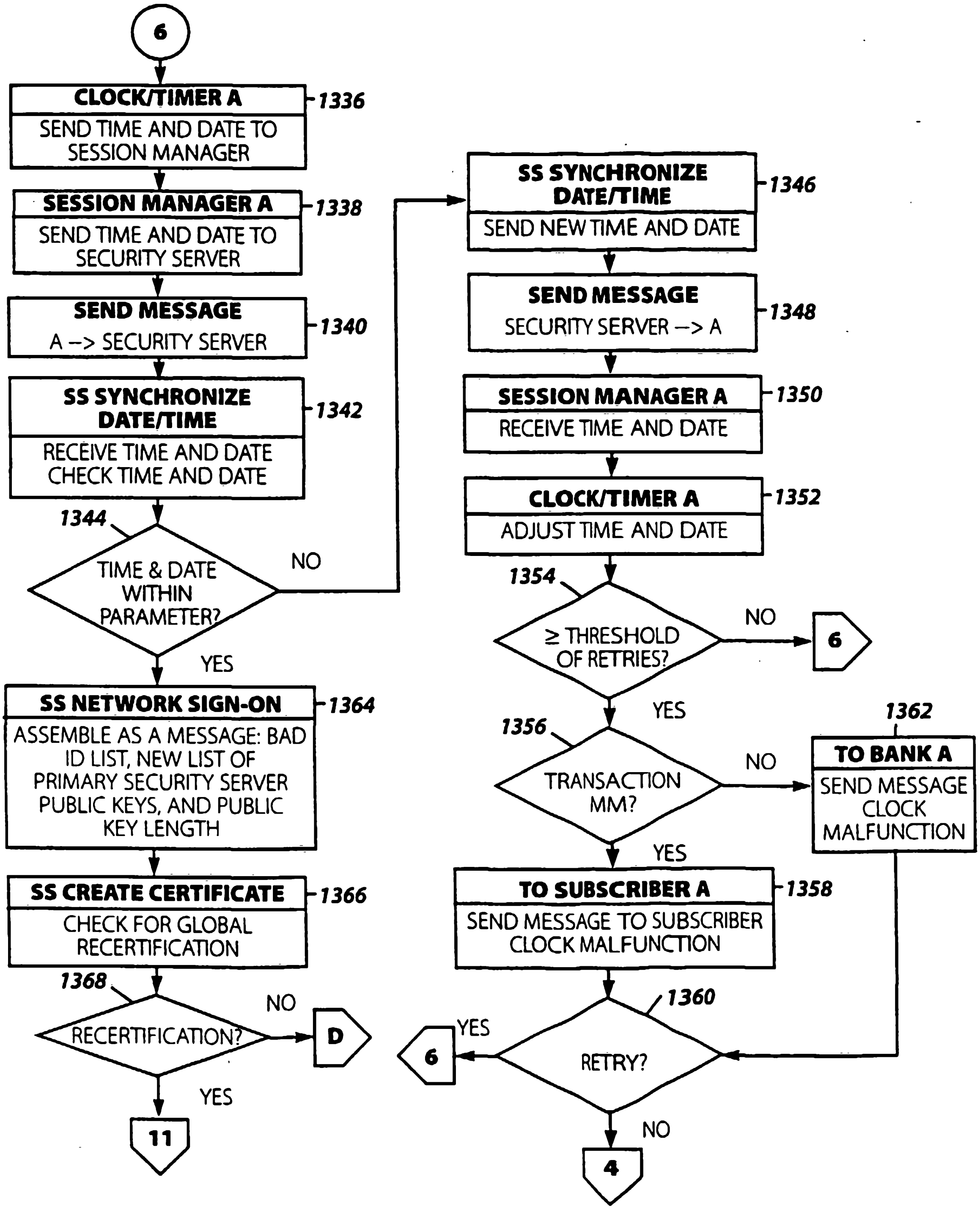

6
CLOCK/TIMER A
SEND TIME AND DATE TO SESSION MANAGER
1336
SESSION MANAGER A
SEND TIME AND DATE TO SECURITY SERVER
1338
SEND MESSAGE
A --> SECURITY SERVER
1340
SS SYNCHRONIZE DATE/TIME
RECEIVE TIME AND DATE
CHECK TIME AND DATE
1342
1344
TIME & DATE WITHIN PARAMETER?
NO
YES
SS NETWORK SIGN-ON
ASSEMBLE AS A MESSAGE: BAD ID LIST, NEW LIST OF PRIMARY SECURITY SERVER PUBLIC KEYS, AND PUBLIC KEY LENGTH
1364
SS CREATE CERTIFICATE
CHECK FOR GLOBAL RECERTIFICATION
1366
1368
RECERTIFICATION?
NO
D
YES
11
SS SYNCHRONIZE DATE/TIME
SEND NEW TIME AND DATE
1346
SEND MESSAGE
SECURITY SERVER --> A
1348
SESSION MANAGER A
RECEIVE TIME AND DATE
1350
CLOCK/TIMER A
ADJUST TIME AND DATE
1352
1354
≥ THRESHOLD OF RETRIES?
NO
6
YES
1356
TRANSACTION MM?
NO
1362
TO BANK A
SEND MESSAGE CLOCK MALFUNCTION
YES
TO SUBSCRIBER A
SEND MESSAGE TO SUBSCRIBER CLOCK MALFUNCTION
1358
1360
RETRY?
YES
6
NO
4

Figure 6F

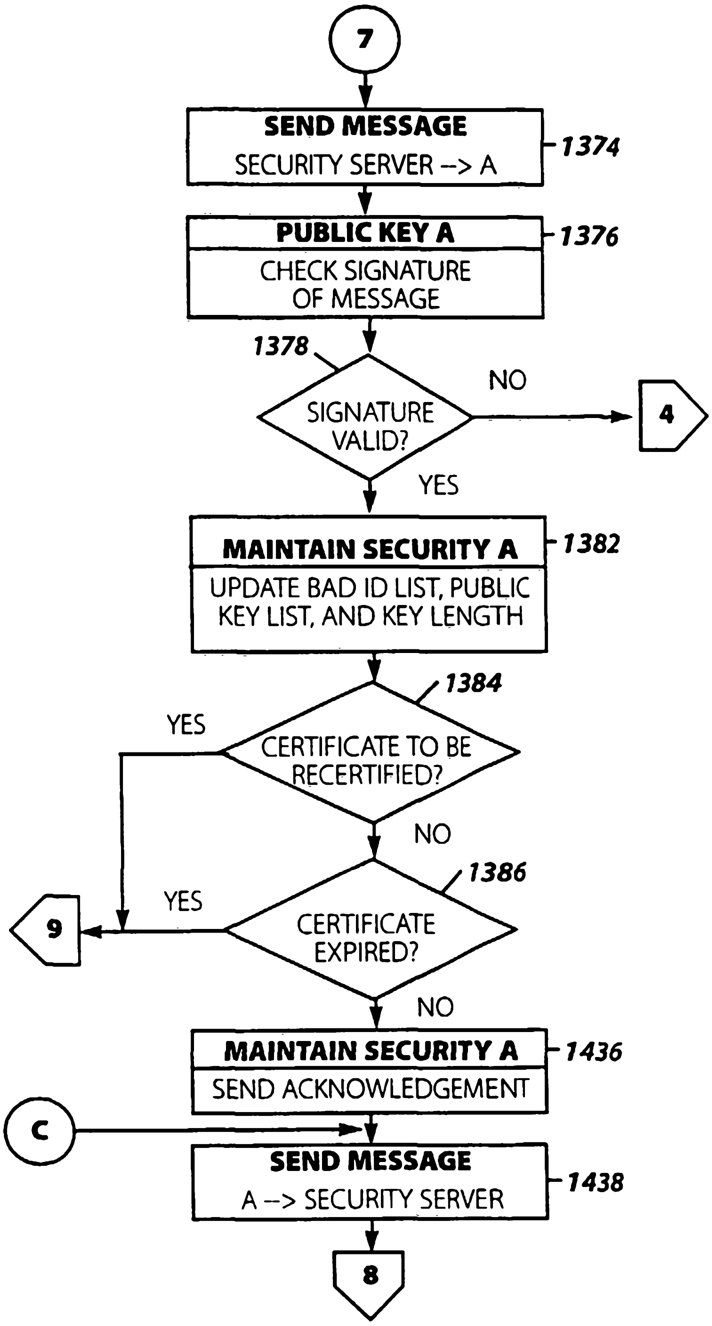

7
SEND MESSAGE
SECURITY SERVER --> A
1374
PUBLIC KEY A
CHECK SIGNATURE OF MESSAGE
1376
1378
SIGNATURE VALID?
NO
4
YES
MAINTAIN SECURITY A
UPDATE BAD ID LIST, PUBLIC KEY LIST, AND KEY LENGTH
1382
1384
YES
CERTIFICATE TO BE RECERTIFIED?
NO
1386
YES
9
CERTIFICATE EXPIRED?
NO
MAINTAIN SECURITY A
SEND ACKNOWLEDGEMENT
1436
C
SEND MESSAGE
A --> SECURITY SERVER
1438
8

Figure 6G

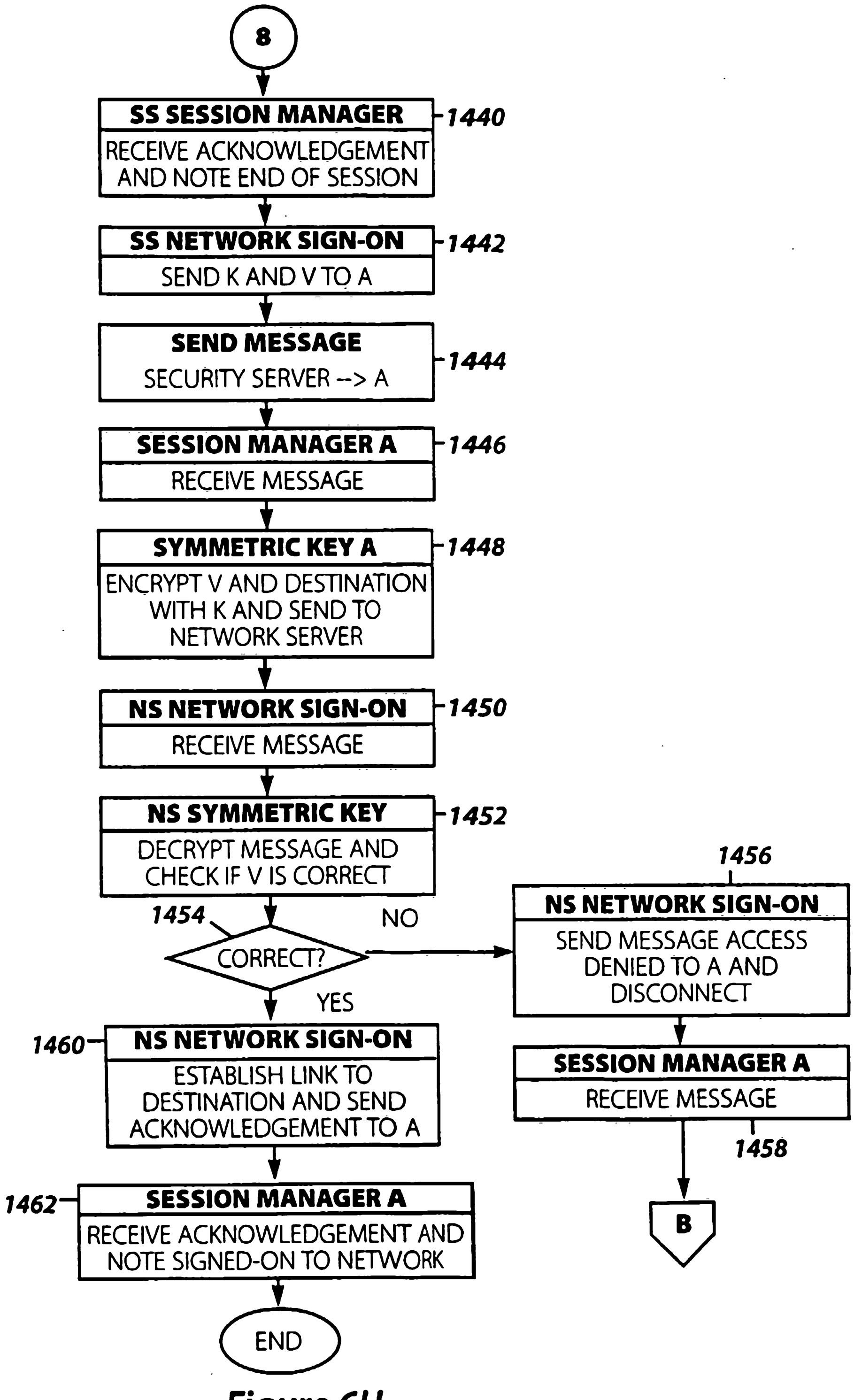

8
SS SESSION MANAGER
RECEIVE ACKNOWLEDGEMENT AND NOTE END OF SESSION
1440
SS NETWORK SIGN-ON
SEND K AND V TO A
1442
SEND MESSAGE
SECURITY SERVER --> A
1444
SESSION MANAGER A
RECEIVE MESSAGE
1446
SYMMETRIC KEY A
ENCRYPT V AND DESTINATION WITH K AND SEND TO NETWORK SERVER
1448
NS NETWORK SIGN-ON
RECEIVE MESSAGE
1450
NS SYMMETRIC KEY
DECRYPT MESSAGE AND CHECK IF V IS CORRECT
1452
1454
CORRECT?
NO
YES
1456
NS NETWORK SIGN-ON
SEND MESSAGE ACCESS DENIED TO A AND DISCONNECT
SESSION MANAGER A
RECEIVE MESSAGE
1458
B
1460
NS NETWORK SIGN-ON
ESTABLISH LINK TO DESTINATION AND SEND ACKNOWLEDGEMENT TO A
1462
SESSION MANAGER A
RECEIVE ACKNOWLEDGEMENT AND NOTE SIGNED-ON TO NETWORK
END

Figure 6H

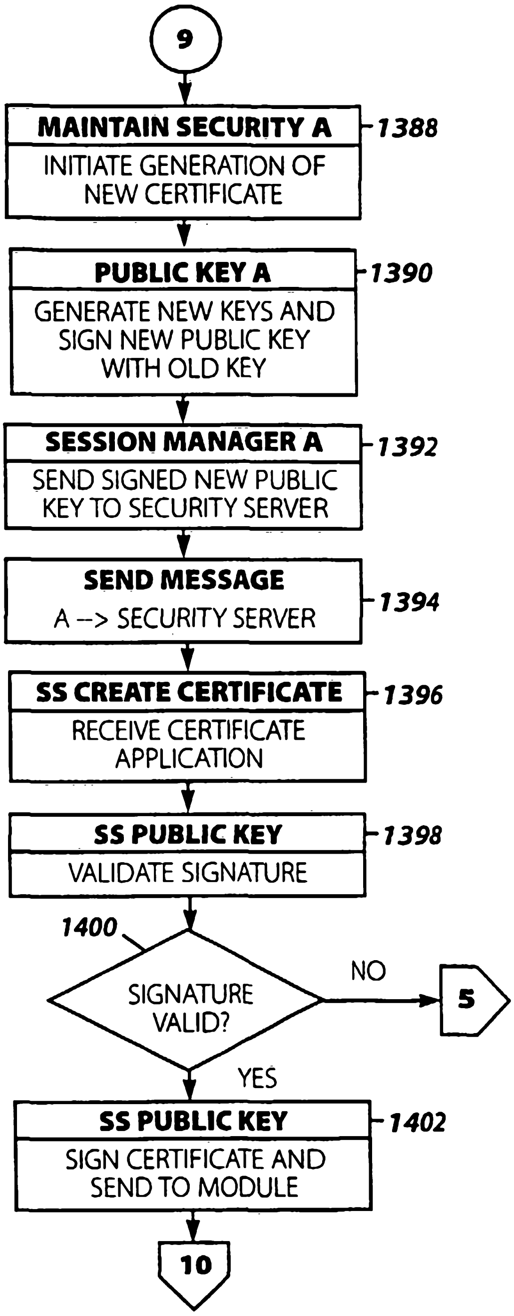

9
MAINTAIN SECURITY A
INITIATE GENERATION OF NEW CERTIFICATE
1388
PUBLIC KEY A
GENERATE NEW KEYS AND SIGN NEW PUBLIC KEY WITH OLD KEY
1390
SESSION MANAGER A
SEND SIGNED NEW PUBLIC KEY TO SECURITY SERVER
1392
SEND MESSAGE
A –> SECURITY SERVER
1394
SS CREATE CERTIFICATE
RECEIVE CERTIFICATE APPLICATION
1396
SS PUBLIC KEY
VALIDATE SIGNATURE
1398
1400
SIGNATURE VALID?
NO
5
YES
SS PUBLIC KEY
SIGN CERTIFICATE AND SEND TO MODULE
1402
10

Figure 6I

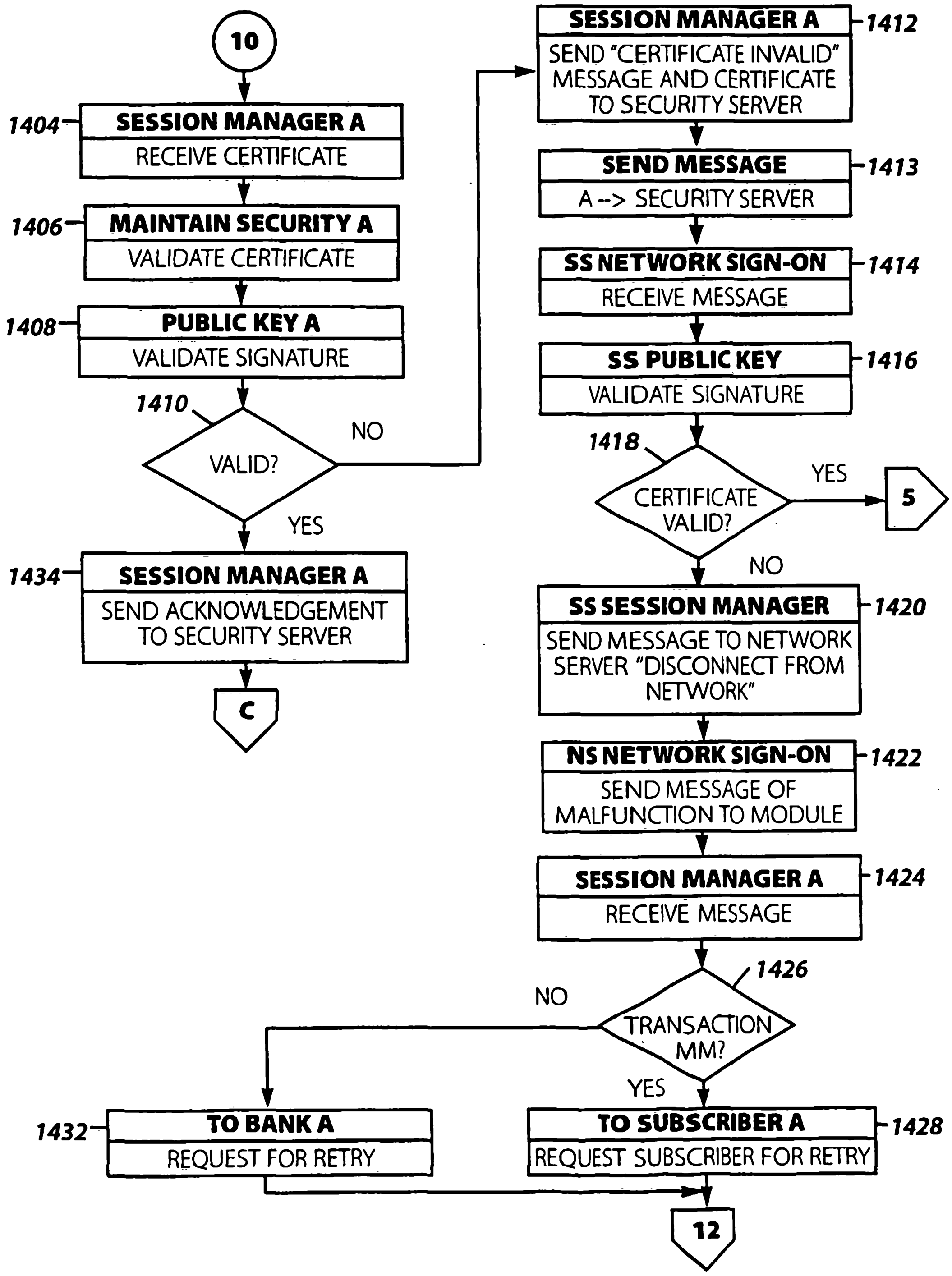

10
1404
SESSION MANAGER A
RECEIVE CERTIFICATE
1406
MAINTAIN SECURITY A
VALIDATE CERTIFICATE
1408
PUBLIC KEY A
VALIDATE SIGNATURE
1410
VALID?
NO
YES
1434
SESSION MANAGER A
SEND ACKNOWLEDGEMENT TO SECURITY SERVER
C
SESSION MANAGER A
SEND "CERTIFICATE INVALID" MESSAGE AND CERTIFICATE TO SECURITY SERVER
1412
SEND MESSAGE
A --> SECURITY SERVER
1413
SS NETWORK SIGN-ON
RECEIVE MESSAGE
1414
SS PUBLIC KEY
VALIDATE SIGNATURE
1416
1418
CERTIFICATE VALID?
YES
5
NO
SS SESSION MANAGER
SEND MESSAGE TO NETWORK SERVER "DISCONNECT FROM NETWORK"
1420
NS NETWORK SIGN-ON
SEND MESSAGE OF MALFUNCTION TO MODULE
1422
SESSION MANAGER A
RECEIVE MESSAGE
1424
1426
NO
TRANSACTION MM?
YES
1432
TO BANK A
REQUEST FOR RETRY
TO SUBSCRIBER A
REQUEST SUBSCRIBER FOR RETRY
1428
12

Figure 6J

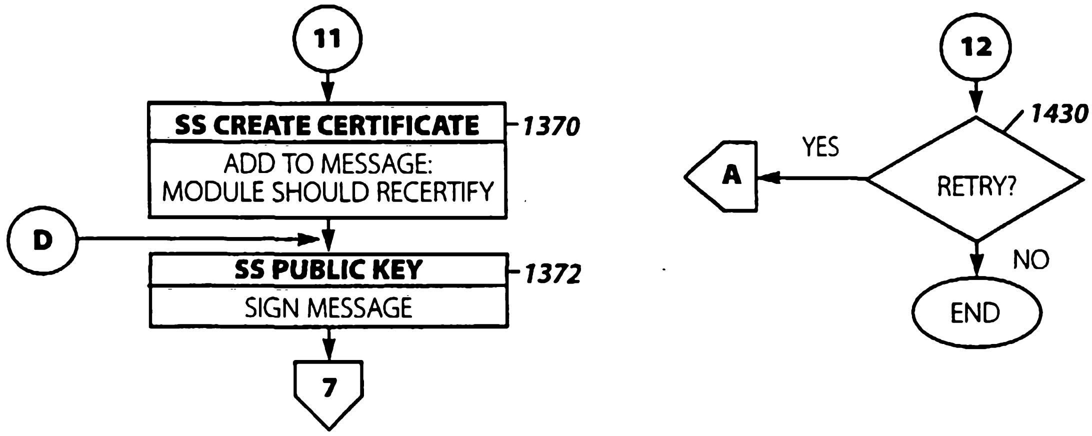

11
SS CREATE CERTIFICATE
ADD TO MESSAGE:
MODULE SHOULD RECERTIFY
1370
D
SS PUBLIC KEY
SIGN MESSAGE
1372
7
12
1430
RETRY?
YES
A
NO
END

Figure 6K

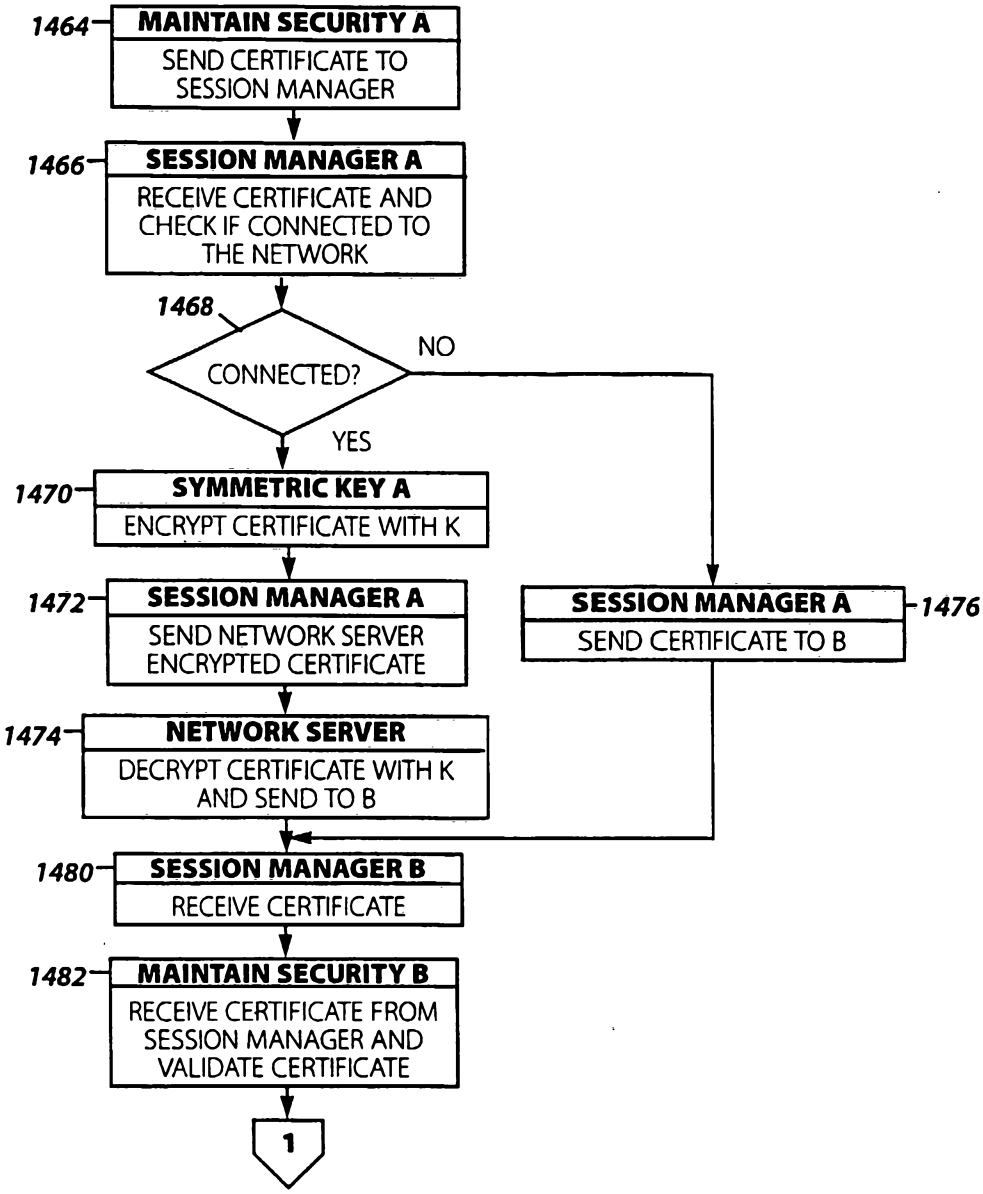

1464
MAINTAIN SECURITY A
SEND CERTIFICATE TO SESSION MANAGER
1466
SESSION MANAGER A
RECEIVE CERTIFICATE AND CHECK IF CONNECTED TO THE NETWORK
1468
CONNECTED?
NO
YES
1470
SYMMETRIC KEY A
ENCRYPT CERTIFICATE WITH K
1472
SESSION MANAGER A
SEND NETWORK SERVER ENCRYPTED CERTIFICATE
1476
SESSION MANAGER A
SEND CERTIFICATE TO B
1474
NETWORK SERVER
DECRYPT CERTIFICATE WITH K AND SEND TO B
1480
SESSION MANAGER B
RECEIVE CERTIFICATE
1482
MAINTAIN SECURITY B
RECEIVE CERTIFICATE FROM SESSION MANAGER AND VALIDATE CERTIFICATE
1

Figure 7A

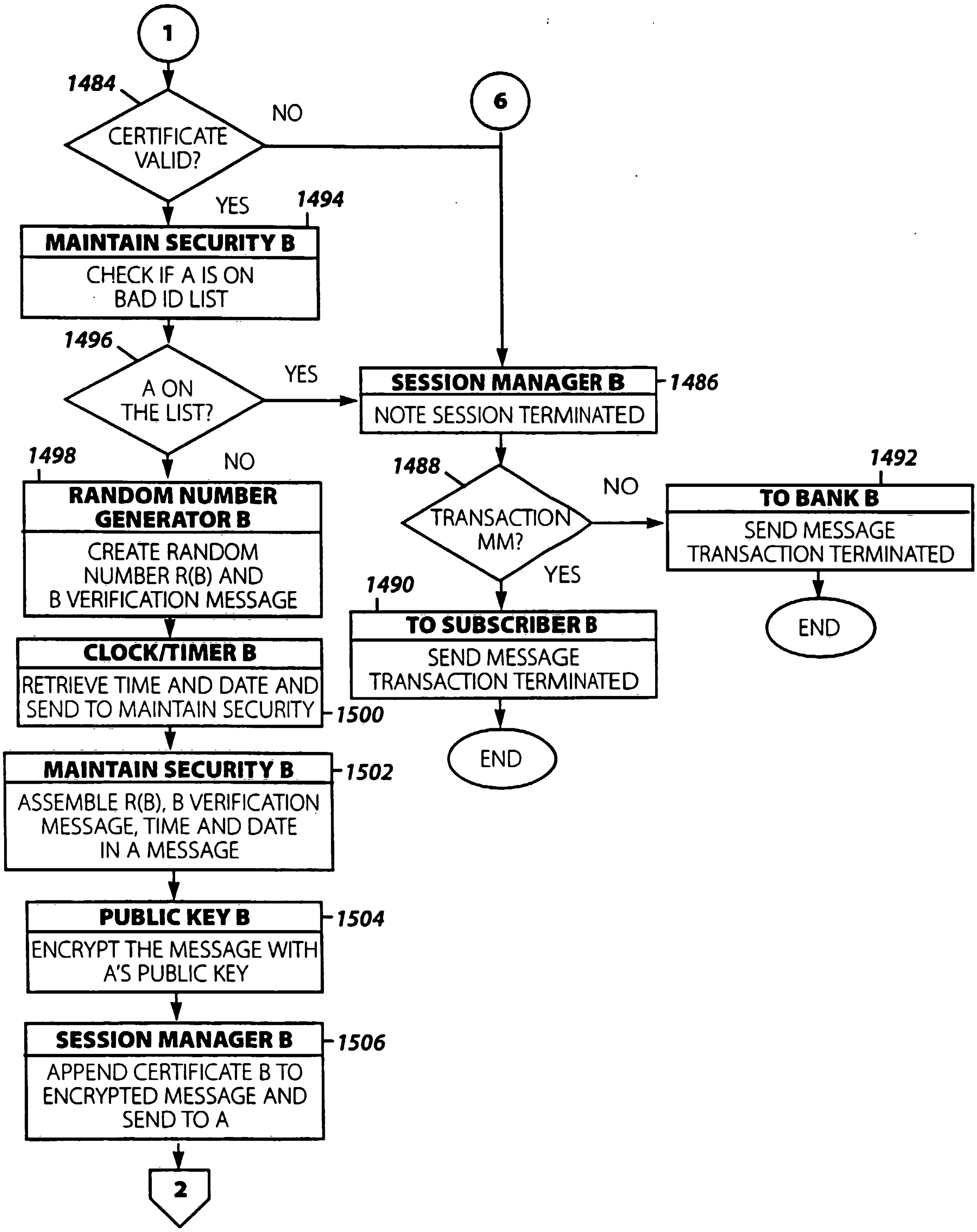

1
1484
CERTIFICATE VALID?
NO
6
YES
1494
MAINTAIN SECURITY B
CHECK IF A IS ON BAD ID LIST
1496
A ON THE LIST?
YES
SESSION MANAGER B
NOTE SESSION TERMINATED
1486
1498
NO
RANDOM NUMBER GENERATOR B
CREATE RANDOM NUMBER R(B) AND B VERIFICATION MESSAGE
1488
TRANSACTION MM?
NO
1492
TO BANK B
SEND MESSAGE TRANSACTION TERMINATED
END
1490
YES
TO SUBSCRIBER B
SEND MESSAGE TRANSACTION TERMINATED
END
CLOCK/TIMER B
RETRIEVE TIME AND DATE AND SEND TO MAINTAIN SECURITY
1500
MAINTAIN SECURITY B
ASSEMBLE R(B), B VERIFICATION MESSAGE, TIME AND DATE IN A MESSAGE
1502
PUBLIC KEY B
ENCRYPT THE MESSAGE WITH A'S PUBLIC KEY
1504
SESSION MANAGER B
APPEND CERTIFICATE B TO ENCRYPTED MESSAGE AND SEND TO A
1506
2

Figure 7B

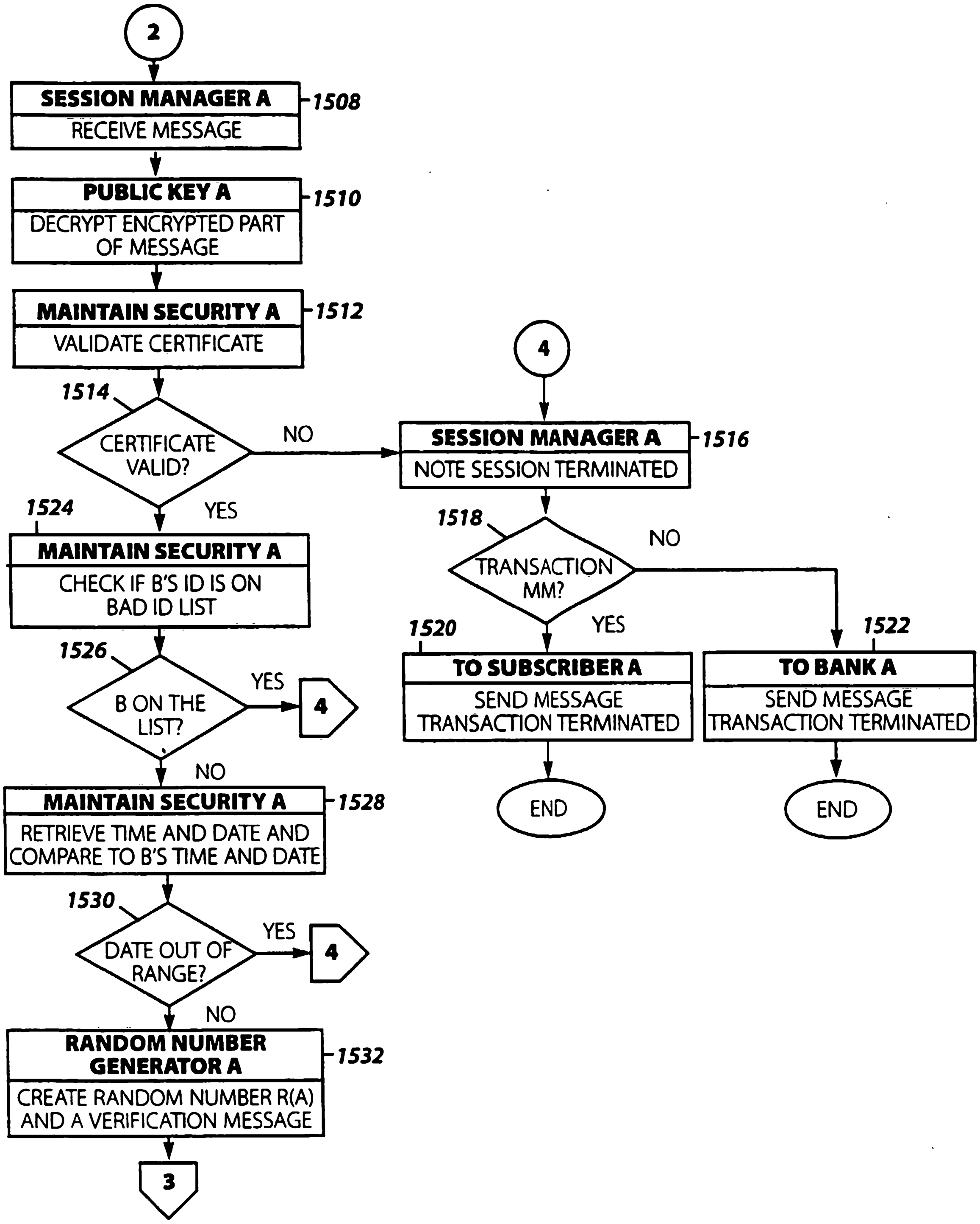

2
SESSION MANAGER A
RECEIVE MESSAGE
1508
PUBLIC KEY A
DECRYPT ENCRYPTED PART OF MESSAGE
1510
MAINTAIN SECURITY A
VALIDATE CERTIFICATE
1512
1514
CERTIFICATE VALID?
NO
YES
4
SESSION MANAGER A
NOTE SESSION TERMINATED
1516
1518
TRANSACTION MM?
NO
YES
1520
TO SUBSCRIBER A
SEND MESSAGE TRANSACTION TERMINATED
1522
TO BANK A
SEND MESSAGE TRANSACTION TERMINATED
END
END
1524
MAINTAIN SECURITY A
CHECK IF B'S ID IS ON BAD ID LIST
1526
B ON THE LIST?
YES
4
NO
MAINTAIN SECURITY A
RETRIEVE TIME AND DATE AND COMPARE TO B'S TIME AND DATE
1528
1530
DATE OUT OF RANGE?
YES
4
NO
RANDOM NUMBER GENERATOR A
CREATE RANDOM NUMBER R(A) AND A VERIFICATION MESSAGE
1532
3

Figure 7C

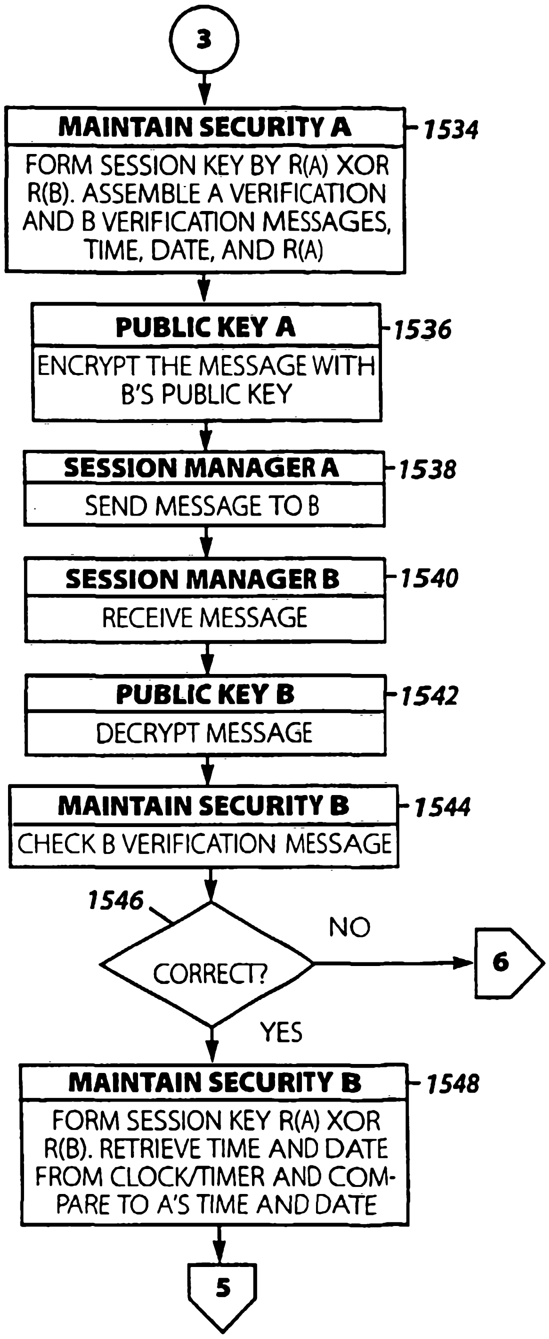

3
MAINTAIN SECURITY A
FORM SESSION KEY BY R(A) XOR R(B). ASSEMBLE A VERIFICATION AND B VERIFICATION MESSAGES, TIME, DATE, AND R(A)
1534
PUBLIC KEY A
ENCRYPT THE MESSAGE WITH B'S PUBLIC KEY
1536
SESSION MANAGER A
SEND MESSAGE TO B
1538
SESSION MANAGER B
RECEIVE MESSAGE
1540
PUBLIC KEY B
DECRYPT MESSAGE
1542
MAINTAIN SECURITY B
CHECK B VERIFICATION MESSAGE
1544
1546
CORRECT?
NO
6
YES
MAINTAIN SECURITY B
FORM SESSION KEY R(A) XOR R(B). RETRIEVE TIME AND DATE FROM CLOCK/TIMER AND COMPARE TO A'S TIME AND DATE
1548
5

Figure 7D

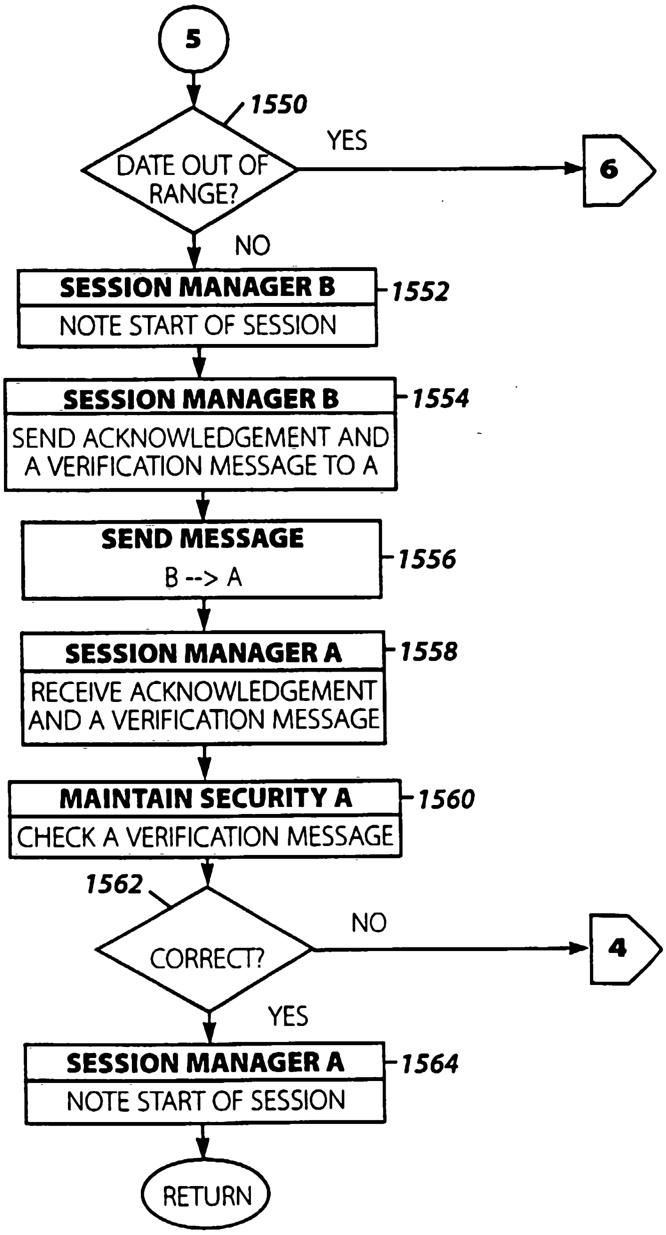

5
1550
DATE OUT OF RANGE?
YES
6
NO
SESSION MANAGER B
NOTE START OF SESSION
1552
SESSION MANAGER B
SEND ACKNOWLEDGEMENT AND A VERIFICATION MESSAGE TO A
1554
SEND MESSAGE
B --> A
1556
SESSION MANAGER A
RECEIVE ACKNOWLEDGEMENT AND A VERIFICATION MESSAGE
1558
MAINTAIN SECURITY A
CHECK A VERIFICATION MESSAGE
1560
1562
CORRECT?
NO
4
YES
SESSION MANAGER A
NOTE START OF SESSION
1564
RETURN

Figure 7E

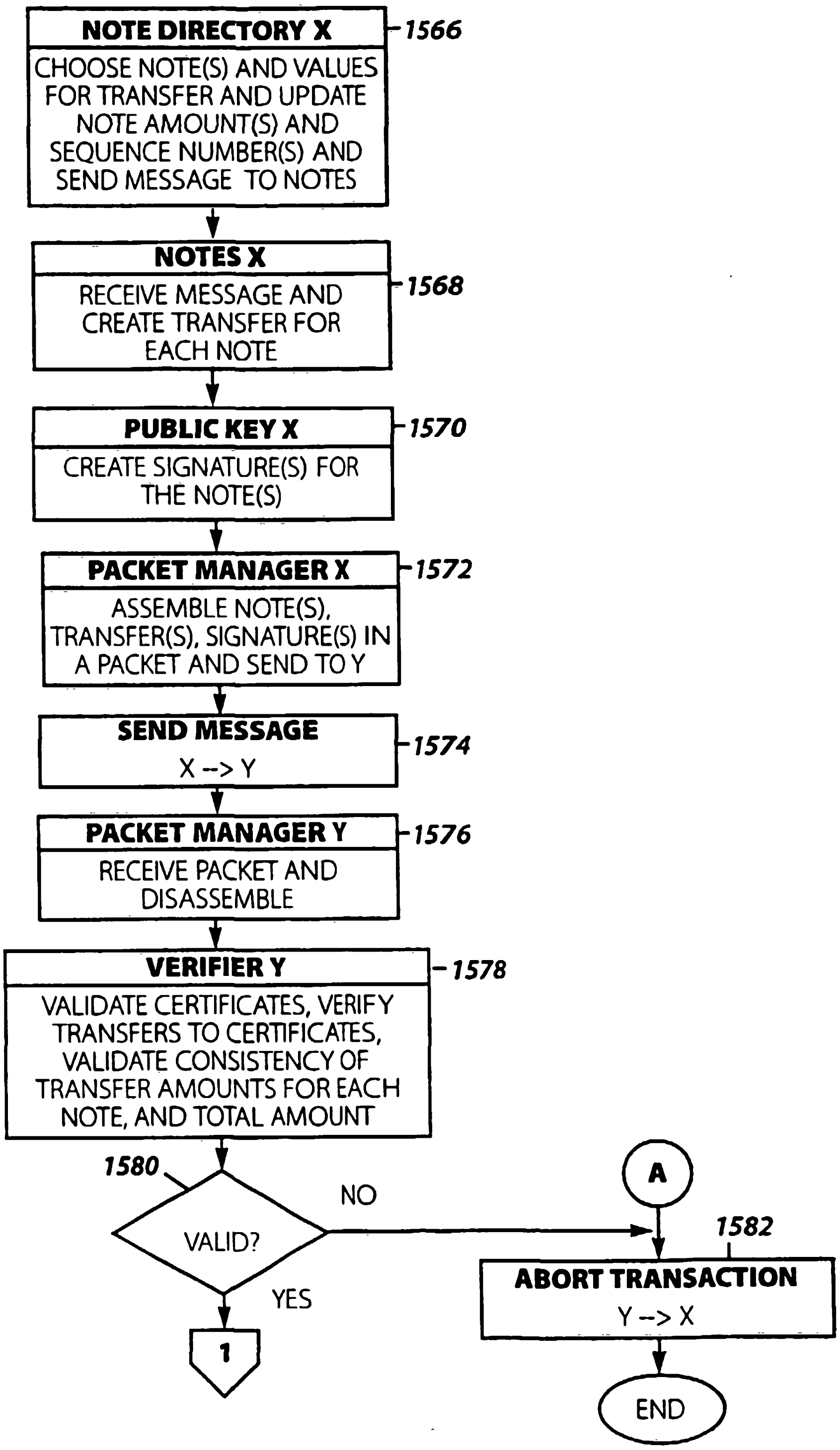

NOTE DIRECTORY X
CHOOSE NOTE(S) AND VALUES FOR TRANSFER AND UPDATE NOTE AMOUNT(S) AND SEQUENCE NUMBER(S) AND SEND MESSAGE TO NOTES
1566
NOTES X
RECEIVE MESSAGE AND CREATE TRANSFER FOR EACH NOTE
1568
PUBLIC KEY X
CREATE SIGNATURE(S) FOR THE NOTE(S)
1570
PACKET MANAGER X
ASSEMBLE NOTE(S), TRANSFER(S), SIGNATURE(S) IN A PACKET AND SEND TO Y
1572
SEND MESSAGE
X --> Y
1574
PACKET MANAGER Y
RECEIVE PACKET AND DISASSEMBLE
1576
VERIFIER Y
VALIDATE CERTIFICATES, VERIFY TRANSFERS TO CERTIFICATES, VALIDATE CONSISTENCY OF TRANSFER AMOUNTS FOR EACH NOTE, AND TOTAL AMOUNT
1578
1580
VALID?
NO
YES
1
A
1582
ABORT TRANSACTION
Y --> X
END

Figure 8A

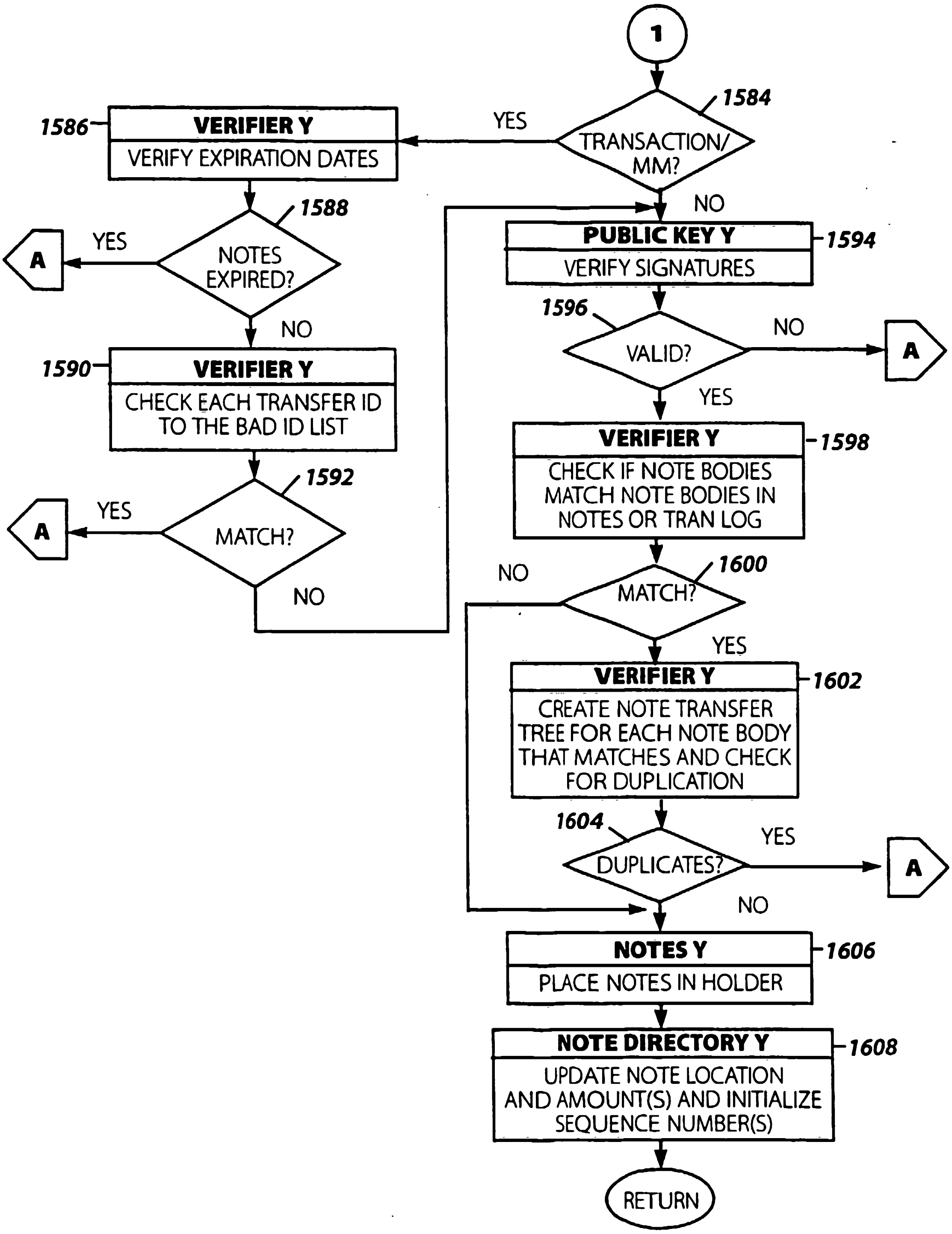

1
1584
TRANSACTION/ MM?
YES
NO
1586
VERIFIER Y
VERIFY EXPIRATION DATES
1588
NOTES EXPIRED?
YES
A
NO
1590
VERIFIER Y
CHECK EACH TRANSFER ID TO THE BAD ID LIST
1592
MATCH?
YES
A
NO
PUBLIC KEY Y
VERIFY SIGNATURES
1594
1596
VALID?
NO
A
YES
VERIFIER Y
CHECK IF NOTE BODIES MATCH NOTE BODIES IN NOTES OR TRAN LOG
1598
1600
MATCH?
NO
YES
VERIFIER Y
CREATE NOTE TRANSFER TREE FOR EACH NOTE BODY THAT MATCHES AND CHECK FOR DUPLICATION
1602
1604
DUPLICATES?
YES
A
NO
NOTES Y
PLACE NOTES IN HOLDER
1606
NOTE DIRECTORY Y
UPDATE NOTE LOCATION AND AMOUNT(S) AND INITIALIZE SEQUENCE NUMBER(S)
1608
RETURN

Figure 8B

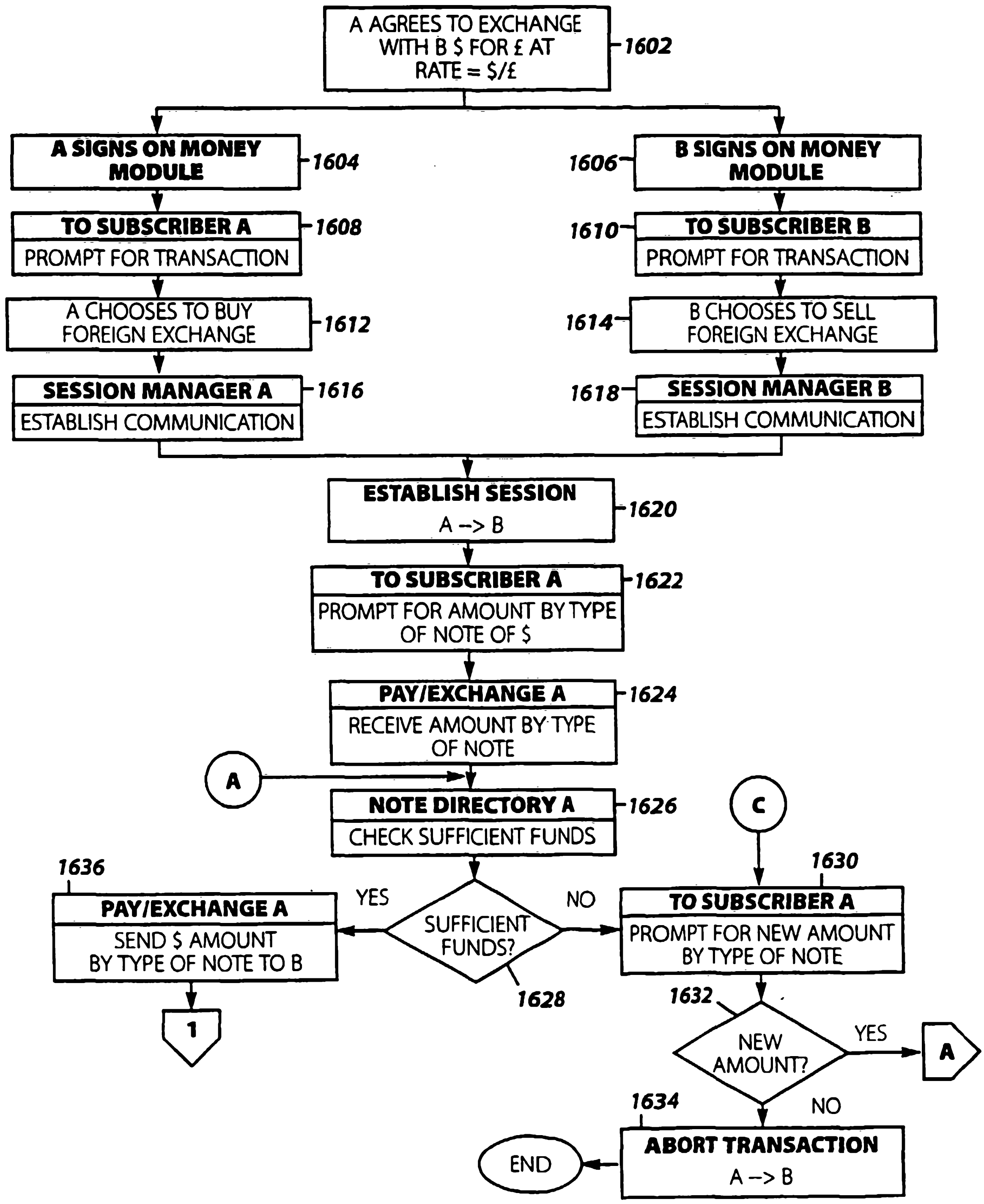

A AGREES TO EXCHANGE WITH B $ FOR £ AT RATE = $/£
1602
A SIGNS ON MONEY MODULE
1604
B SIGNS ON MONEY MODULE
1606
TO SUBSCRIBER A
PROMPT FOR TRANSACTION
1608
TO SUBSCRIBER B
PROMPT FOR TRANSACTION
1610
A CHOOSES TO BUY FOREIGN EXCHANGE
1612
B CHOOSES TO SELL FOREIGN EXCHANGE
1614
SESSION MANAGER A
ESTABLISH COMMUNICATION
1616
SESSION MANAGER B
ESTABLISH COMMUNICATION
1618
ESTABLISH SESSION
A -> B
1620
TO SUBSCRIBER A
PROMPT FOR AMOUNT BY TYPE OF NOTE OF $
1622
PAY/EXCHANGE A
RECEIVE AMOUNT BY TYPE OF NOTE
1624
A
NOTE DIRECTORY A
CHECK SUFFICIENT FUNDS
1626
C
SUFFICIENT FUNDS?
1628
YES
NO
PAY/EXCHANGE A
SEND $ AMOUNT BY TYPE OF NOTE TO B
1636
1
TO SUBSCRIBER A
PROMPT FOR NEW AMOUNT BY TYPE OF NOTE
1630
NEW AMOUNT?
1632
YES
A
NO
ABORT TRANSACTION
A -> B
1634
END

Figure 9A

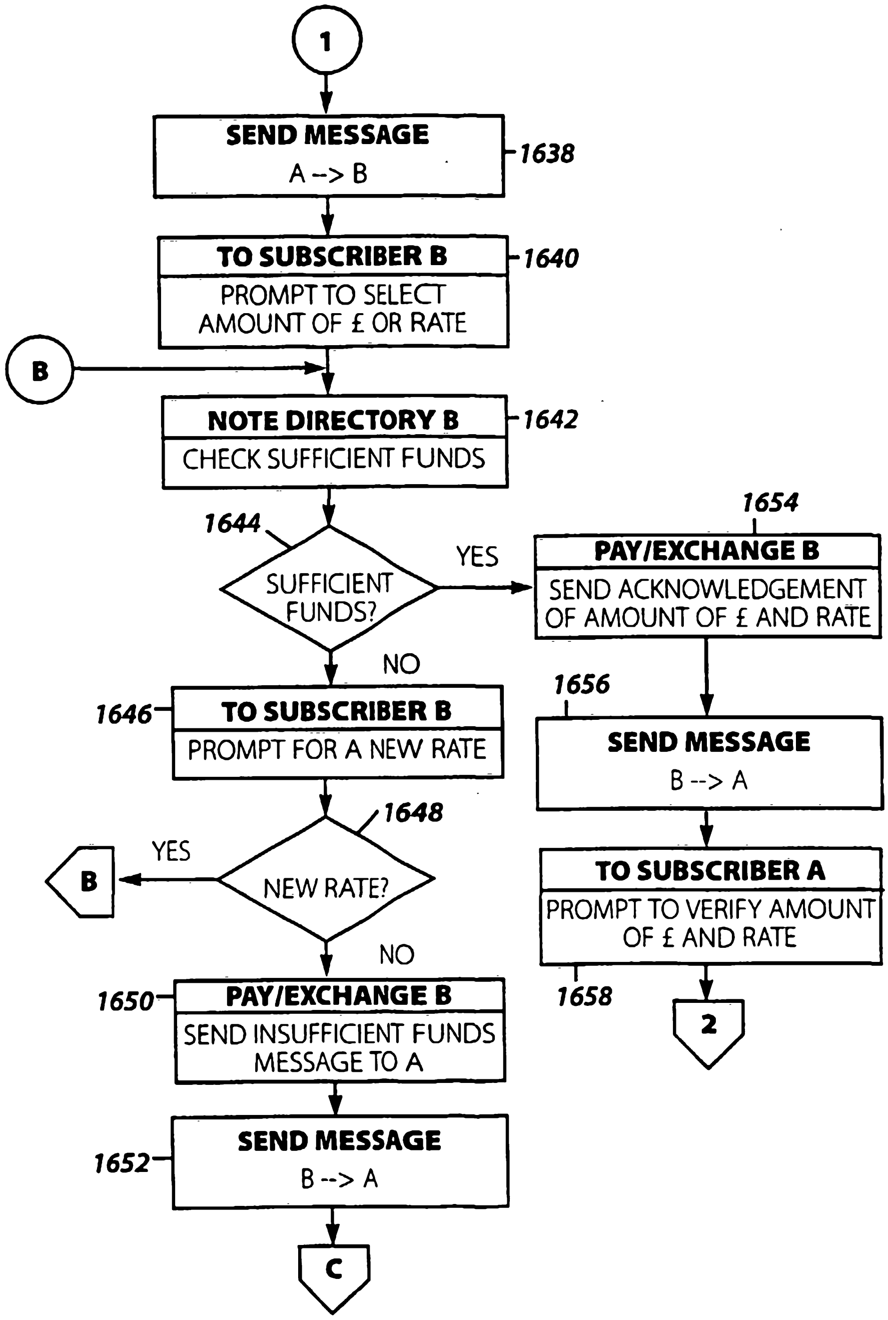

1
SEND MESSAGE
A --> B
1638
TO SUBSCRIBER B
PROMPT TO SELECT AMOUNT OF £ OR RATE
1640
B
NOTE DIRECTORY B
CHECK SUFFICIENT FUNDS
1642
1644
SUFFICIENT FUNDS?
YES
NO
1654
PAY/EXCHANGE B
SEND ACKNOWLEDGEMENT OF AMOUNT OF £ AND RATE
1646
TO SUBSCRIBER B
PROMPT FOR A NEW RATE
1656
SEND MESSAGE
B --> A
1648
NEW RATE?
YES
B
NO
TO SUBSCRIBER A
PROMPT TO VERIFY AMOUNT OF £ AND RATE
1658
2
1650
PAY/EXCHANGE B
SEND INSUFFICIENT FUNDS MESSAGE TO A
1652
SEND MESSAGE
B --> A
C

Figure 9B

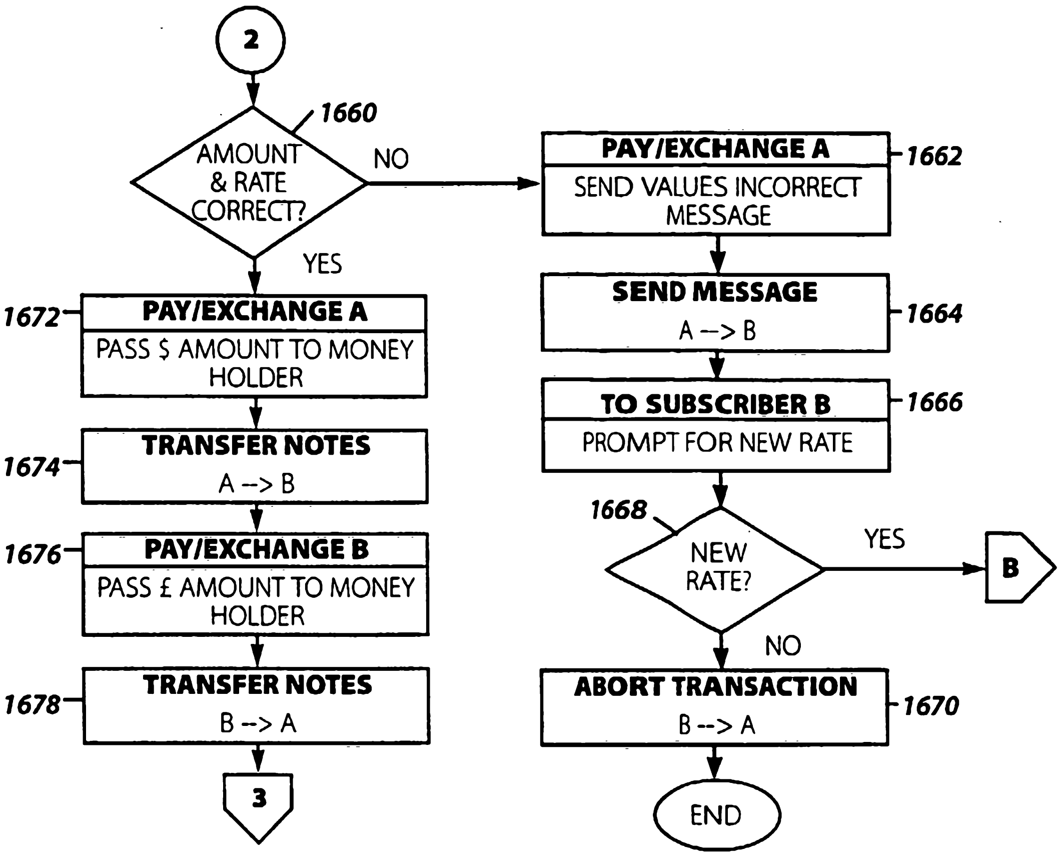

2
1660
AMOUNT & RATE CORRECT?
NO
YES
PAY/EXCHANGE A
SEND VALUES INCORRECT MESSAGE
1662
1672
PAY/EXCHANGE A
PASS $ AMOUNT TO MONEY HOLDER
SEND MESSAGE
A --> B
1664
TO SUBSCRIBER B
PROMPT FOR NEW RATE
1666
1674
TRANSFER NOTES
A --> B
1668
NEW RATE?
YES
B
1676
PAY/EXCHANGE B
PASS £ AMOUNT TO MONEY HOLDER
NO
1678
TRANSFER NOTES
B --> A
ABORT TRANSACTION
B --> A
1670
3
END

Figure 9C

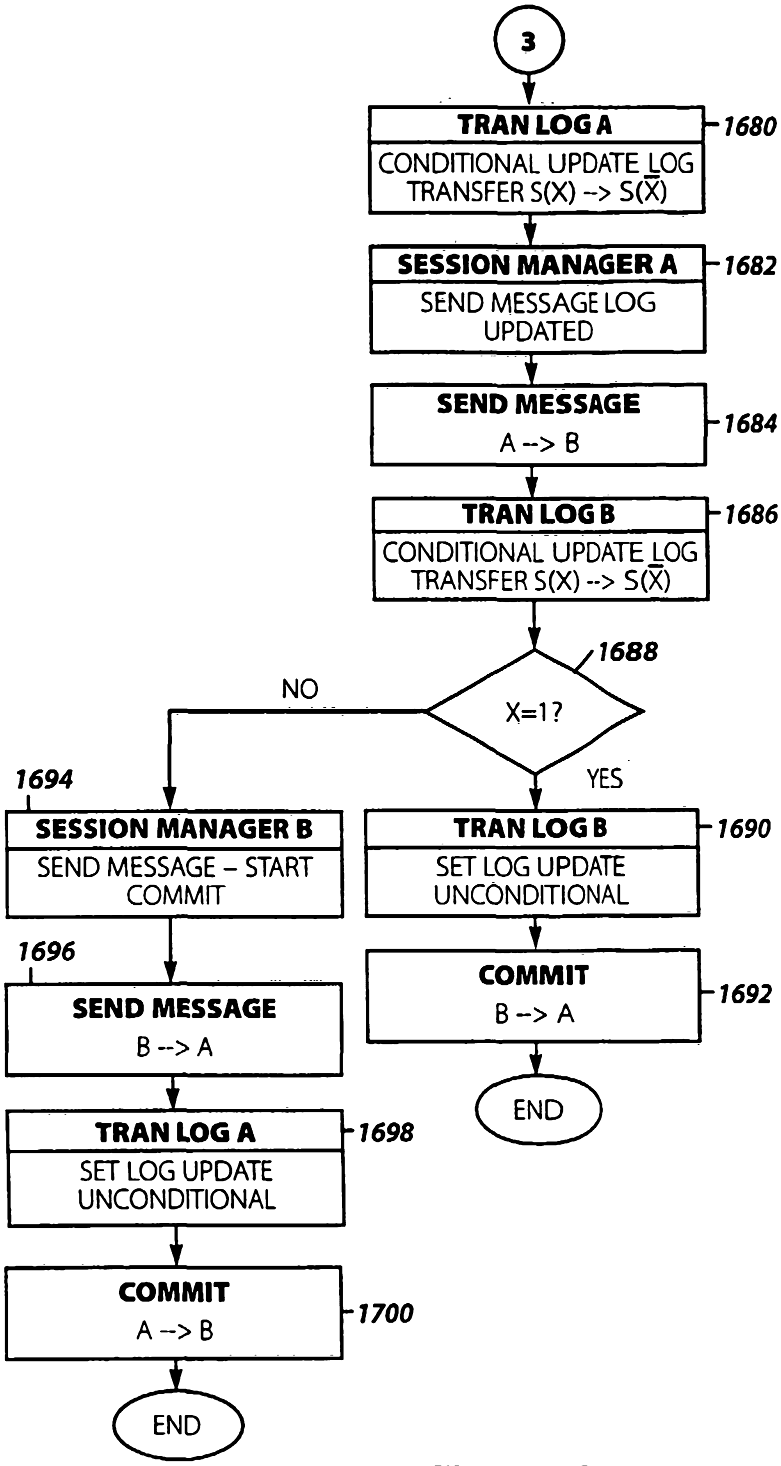

3
TRAN LOG A
CONDITIONAL UPDATE LOG
TRANSFER S(X) --> S(X̄)
1680
SESSION MANAGER A
SEND MESSAGE LOG
UPDATED
1682
SEND MESSAGE
A --> B
1684
TRAN LOG B
CONDITIONAL UPDATE LOG
TRANSFER S(X) --> S(X̄)
1686
1688
X=1?
NO
YES
1694
SESSION MANAGER B
SEND MESSAGE - START
COMMIT
1696
SEND MESSAGE
B --> A
TRAN LOG A
SET LOG UPDATE
UNCONDITIONAL
1698
COMMIT
A --> B
1700
END
TRAN LOG B
SET LOG UPDATE
UNCONDITIONAL
1690
COMMIT
B --> A
1692
END

Figure 9D

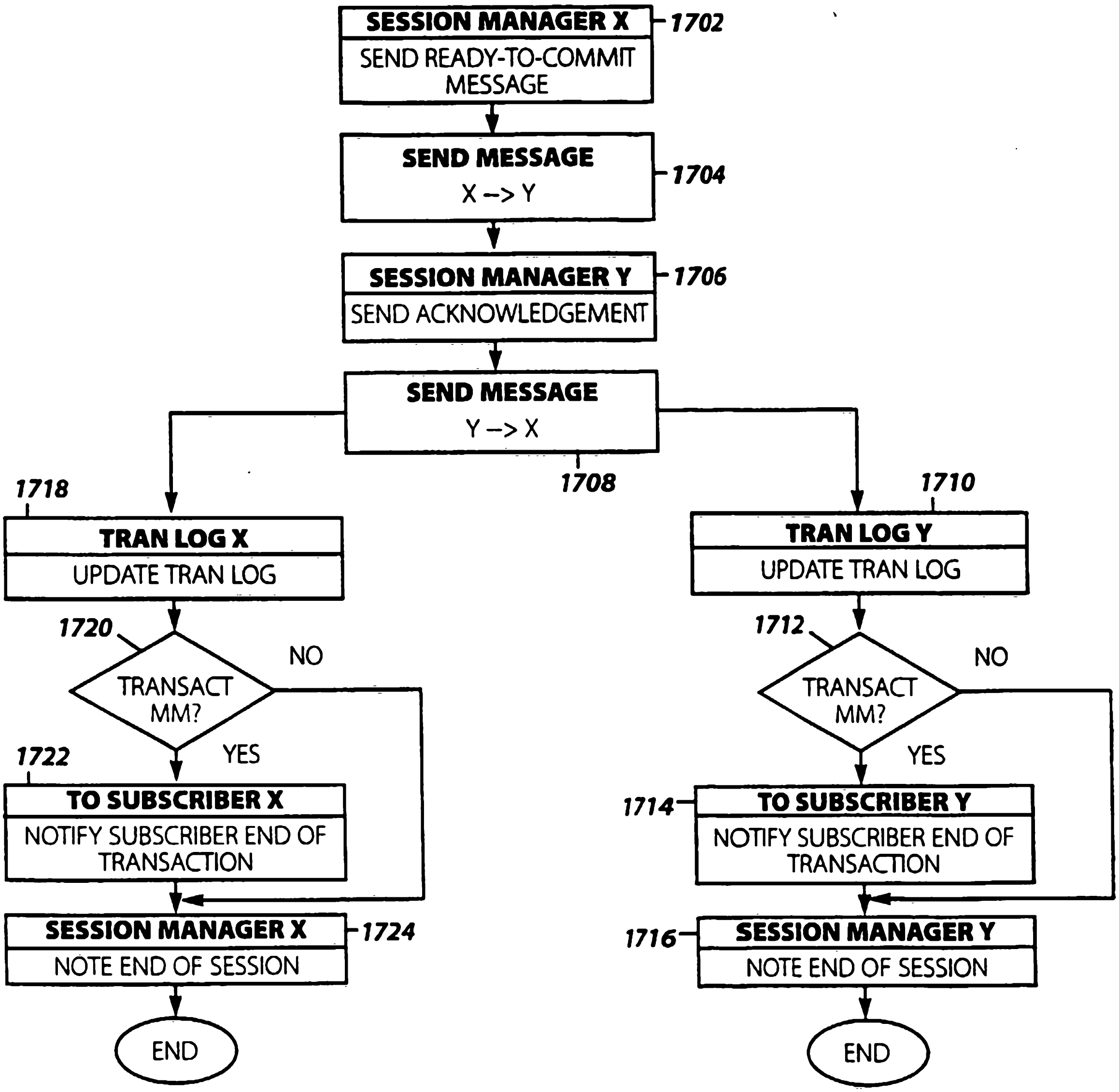

SESSION MANAGER X
SEND READY-TO-COMMIT MESSAGE
1702
SEND MESSAGE
X --> Y
1704
SESSION MANAGER Y
SEND ACKNOWLEDGEMENT
1706
SEND MESSAGE
Y --> X
1708
1718
TRAN LOG X
UPDATE TRAN LOG
1710
TRAN LOG Y
UPDATE TRAN LOG
1720
TRANSACT MM?
NO
YES
1712
TRANSACT MM?
NO
YES
1722
TO SUBSCRIBER X
NOTIFY SUBSCRIBER END OF TRANSACTION
1714
TO SUBSCRIBER Y
NOTIFY SUBSCRIBER END OF TRANSACTION
SESSION MANAGER X
NOTE END OF SESSION
1724
1716
SESSION MANAGER Y
NOTE END OF SESSION
END
END

Figure 10

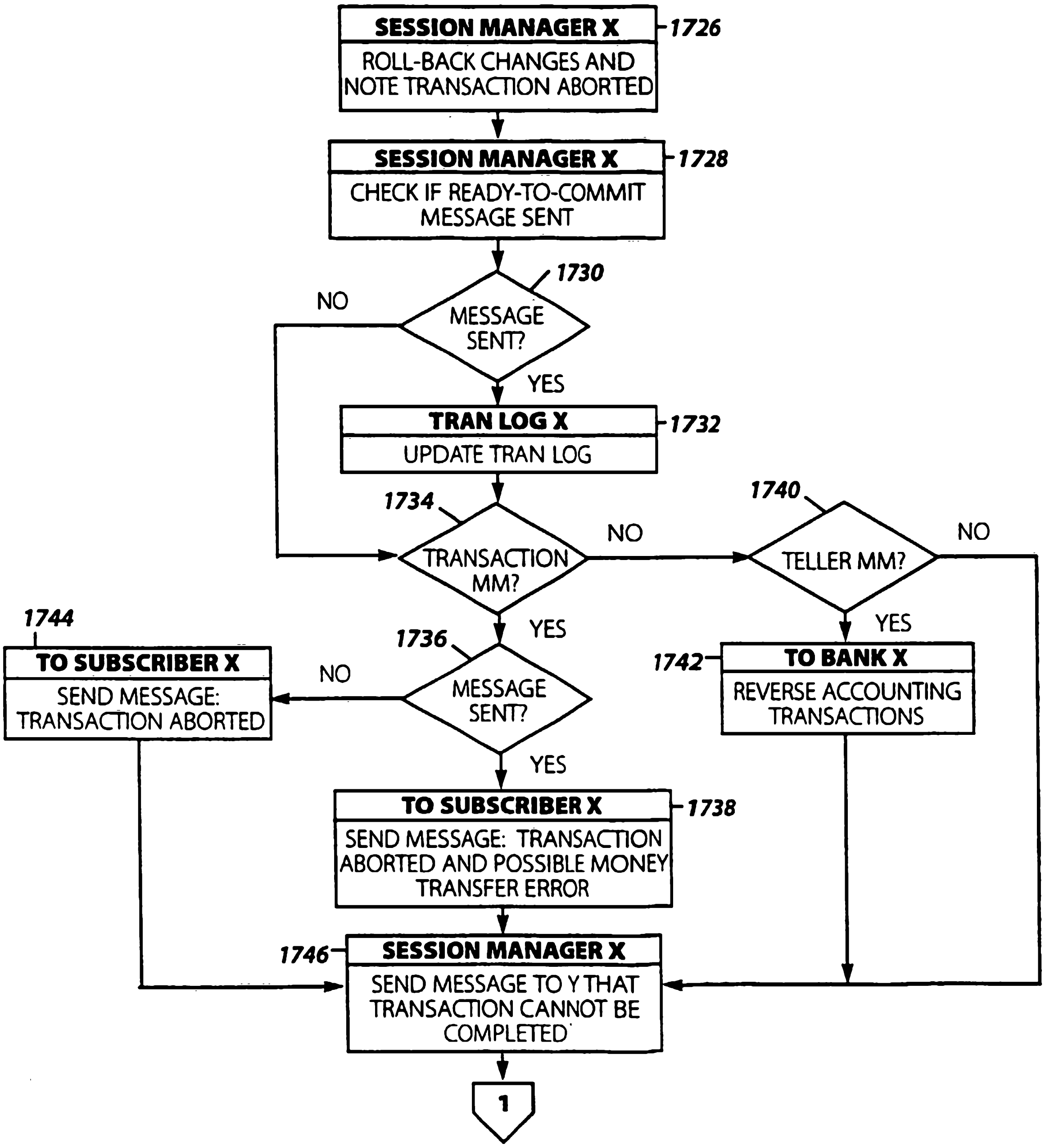

SESSION MANAGER X
ROLL-BACK CHANGES AND NOTE TRANSACTION ABORTED
1726
SESSION MANAGER X
CHECK IF READY-TO-COMMIT MESSAGE SENT
1728
MESSAGE SENT?
1730
NO
YES
TRAN LOG X
UPDATE TRAN LOG
1732
TRANSACTION MM?
1734
NO
YES
TELLER MM?
1740
NO
YES
TO BANK X
REVERSE ACCOUNTING TRANSACTIONS
1742
MESSAGE SENT?
1736
NO
YES
TO SUBSCRIBER X
SEND MESSAGE: TRANSACTION ABORTED
1744
TO SUBSCRIBER X
SEND MESSAGE: TRANSACTION ABORTED AND POSSIBLE MONEY TRANSFER ERROR
1738
SESSION MANAGER X
SEND MESSAGE TO Y THAT TRANSACTION CANNOT BE COMPLETED
1746
1

Figure 11A

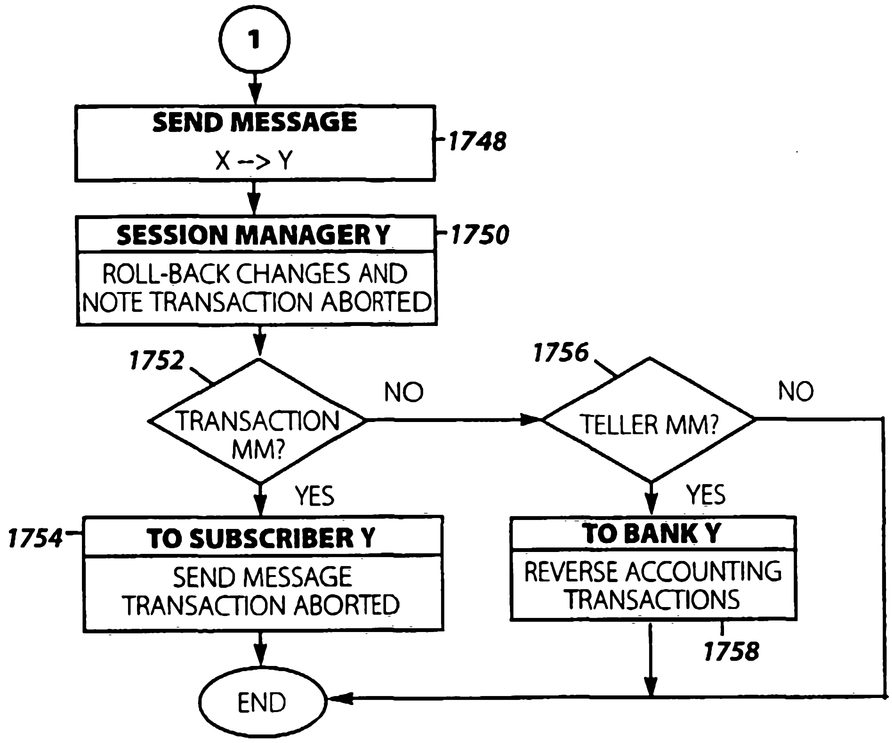

1
SEND MESSAGE
X --> Y
1748
SESSION MANAGER Y
ROLL-BACK CHANGES AND NOTE TRANSACTION ABORTED
1750
1752
TRANSACTION MM?
NO
YES
1754
TO SUBSCRIBER Y
SEND MESSAGE TRANSACTION ABORTED
1756
TELLER MM?
NO
YES
TO BANK Y
REVERSE ACCOUNTING TRANSACTIONS
1758
END

Figure 11B

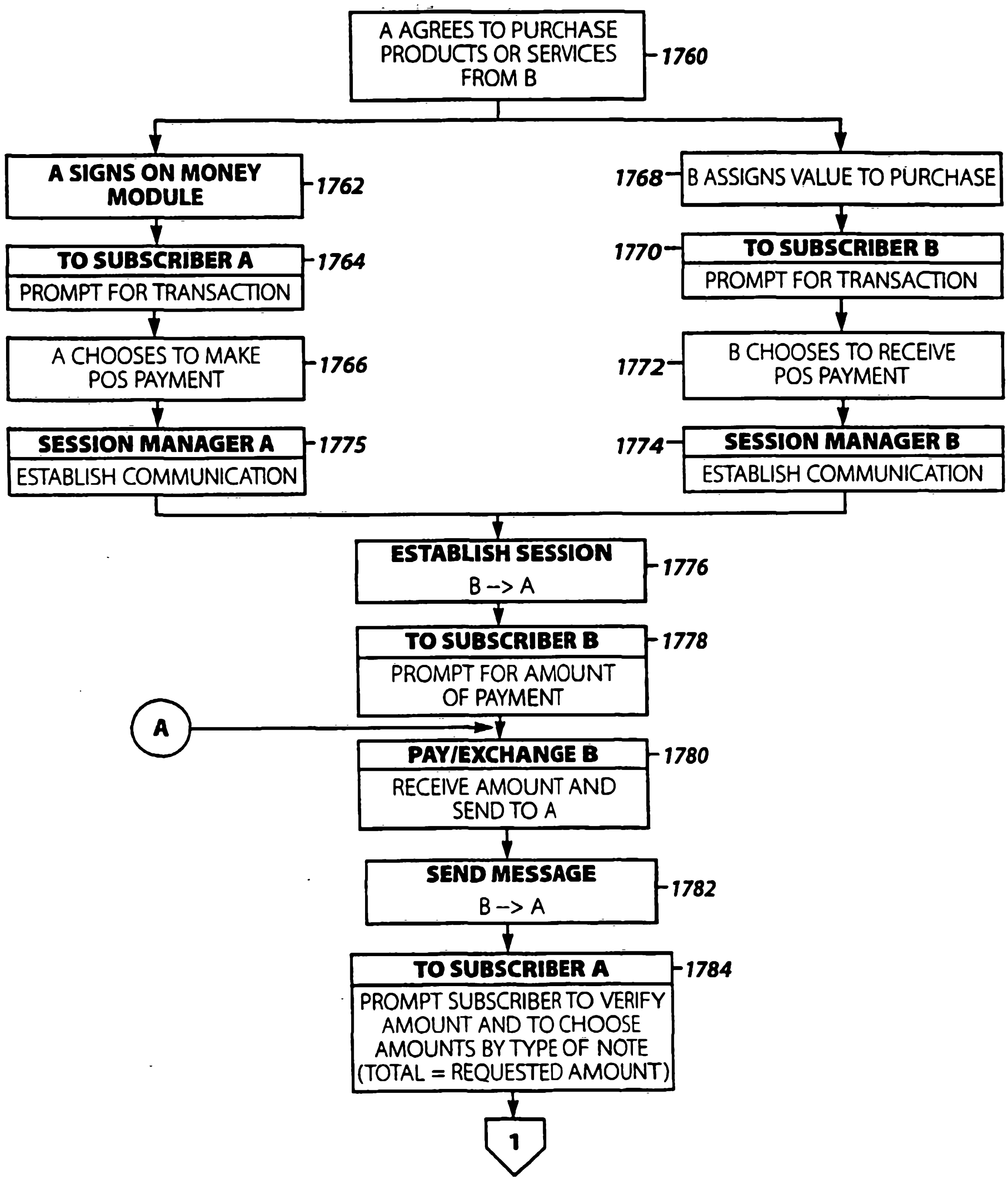

A AGREES TO PURCHASE PRODUCTS OR SERVICES FROM B
1760
A SIGNS ON MONEY MODULE
1762
TO SUBSCRIBER A
PROMPT FOR TRANSACTION
1764
A CHOOSES TO MAKE POS PAYMENT
1766
SESSION MANAGER A
ESTABLISH COMMUNICATION
1775
B ASSIGNS VALUE TO PURCHASE
1768
TO SUBSCRIBER B
PROMPT FOR TRANSACTION
1770
B CHOOSES TO RECEIVE POS PAYMENT
1772
SESSION MANAGER B
ESTABLISH COMMUNICATION
1774
ESTABLISH SESSION
B --> A
1776
TO SUBSCRIBER B
PROMPT FOR AMOUNT OF PAYMENT
1778
A
PAY/EXCHANGE B
RECEIVE AMOUNT AND SEND TO A
1780
SEND MESSAGE
B --> A
1782
TO SUBSCRIBER A
PROMPT SUBSCRIBER TO VERIFY AMOUNT AND TO CHOOSE AMOUNTS BY TYPE OF NOTE (TOTAL = REQUESTED AMOUNT)
1784
1

Figure 12A

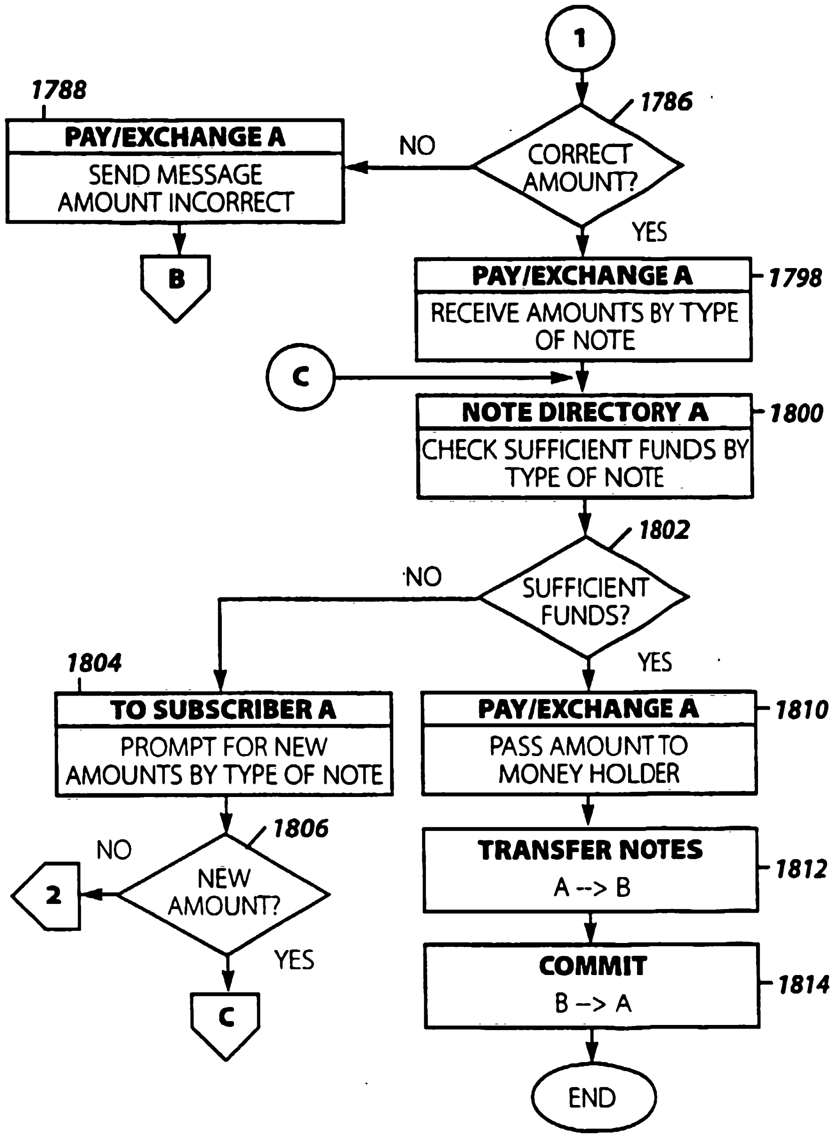

1
1786
CORRECT AMOUNT?
NO
YES
1788
PAY/EXCHANGE A
SEND MESSAGE AMOUNT INCORRECT
B
PAY/EXCHANGE A
RECEIVE AMOUNTS BY TYPE OF NOTE
1798
C
NOTE DIRECTORY A
CHECK SUFFICIENT FUNDS BY TYPE OF NOTE
1800
1802
SUFFICIENT FUNDS?
NO
YES
1804
TO SUBSCRIBER A
PROMPT FOR NEW AMOUNTS BY TYPE OF NOTE
PAY/EXCHANGE A
PASS AMOUNT TO MONEY HOLDER
1810
1806
NEW AMOUNT?
NO
YES
2
C
TRANSFER NOTES
A --> B
1812
COMMIT
B --> A
1814
END

Figure 12B

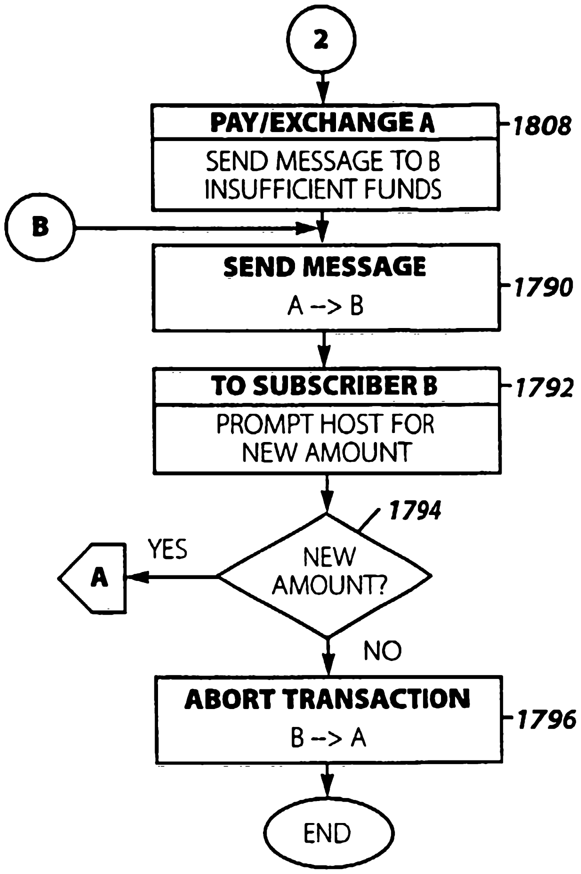

2
PAY/EXCHANGE A
SEND MESSAGE TO B INSUFFICIENT FUNDS
1808
B
SEND MESSAGE
A --> B
1790
TO SUBSCRIBER B
PROMPT HOST FOR NEW AMOUNT
1792
1794
NEW AMOUNT?
YES
A
NO
ABORT TRANSACTION
B --> A
1796
END

Figure 12C

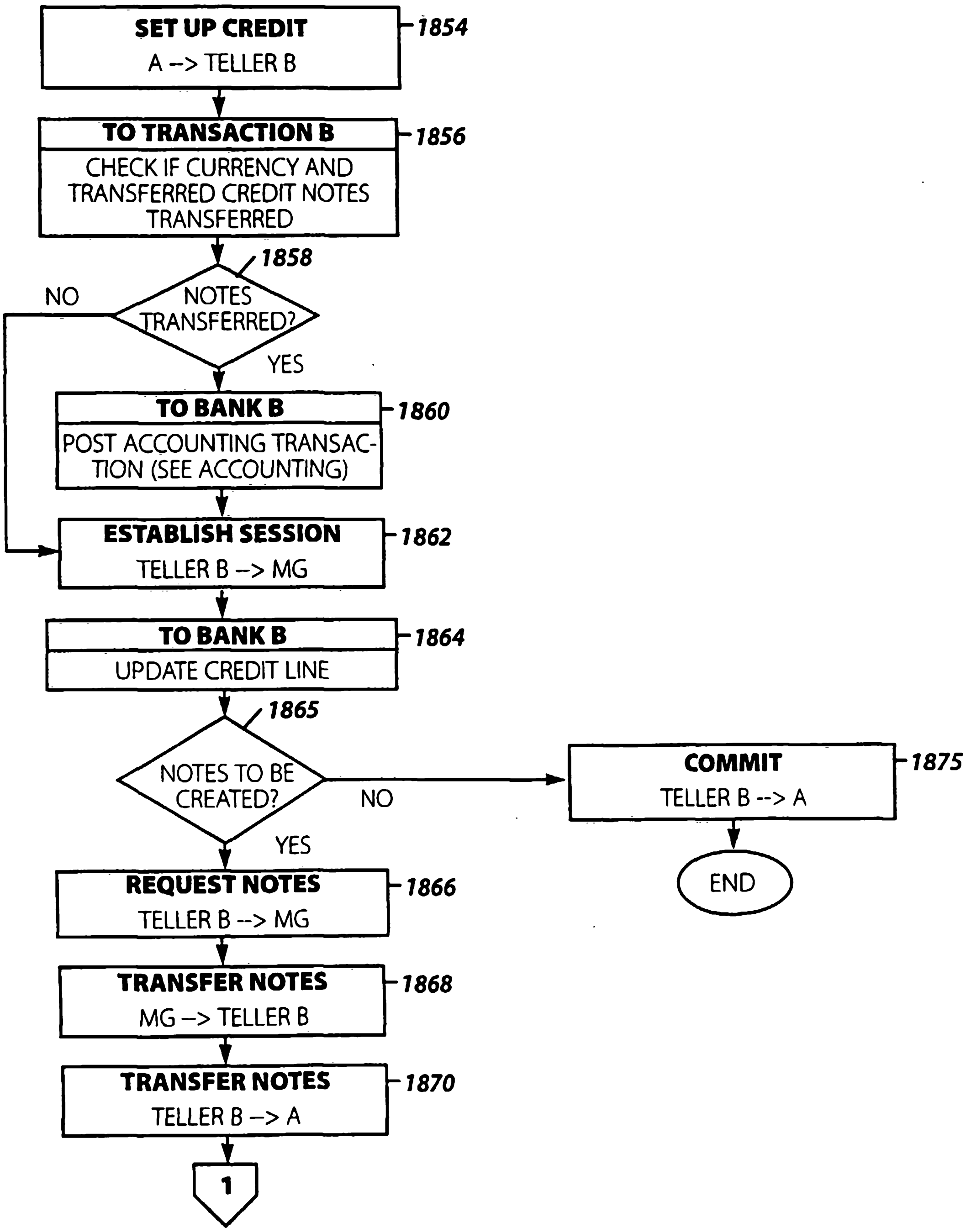

SET UP CREDIT
A --> TELLER B
1854
TO TRANSACTION B
CHECK IF CURRENCY AND TRANSFERRED CREDIT NOTES TRANSFERRED
1856
1858
NOTES TRANSFERRED?
NO
YES
TO BANK B
POST ACCOUNTING TRANSACTION (SEE ACCOUNTING)
1860
ESTABLISH SESSION
TELLER B --> MG
1862
TO BANK B
UPDATE CREDIT LINE
1864
1865
NOTES TO BE CREATED?
NO
YES
COMMIT
TELLER B --> A
1875
END
REQUEST NOTES
TELLER B --> MG
1866
TRANSFER NOTES
MG --> TELLER B
1868
TRANSFER NOTES
TELLER B --> A
1870
1

Figure 13A

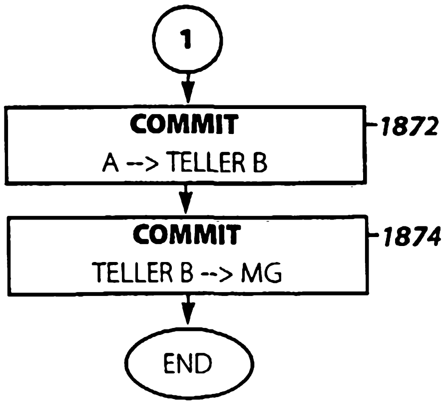


Figure 13B

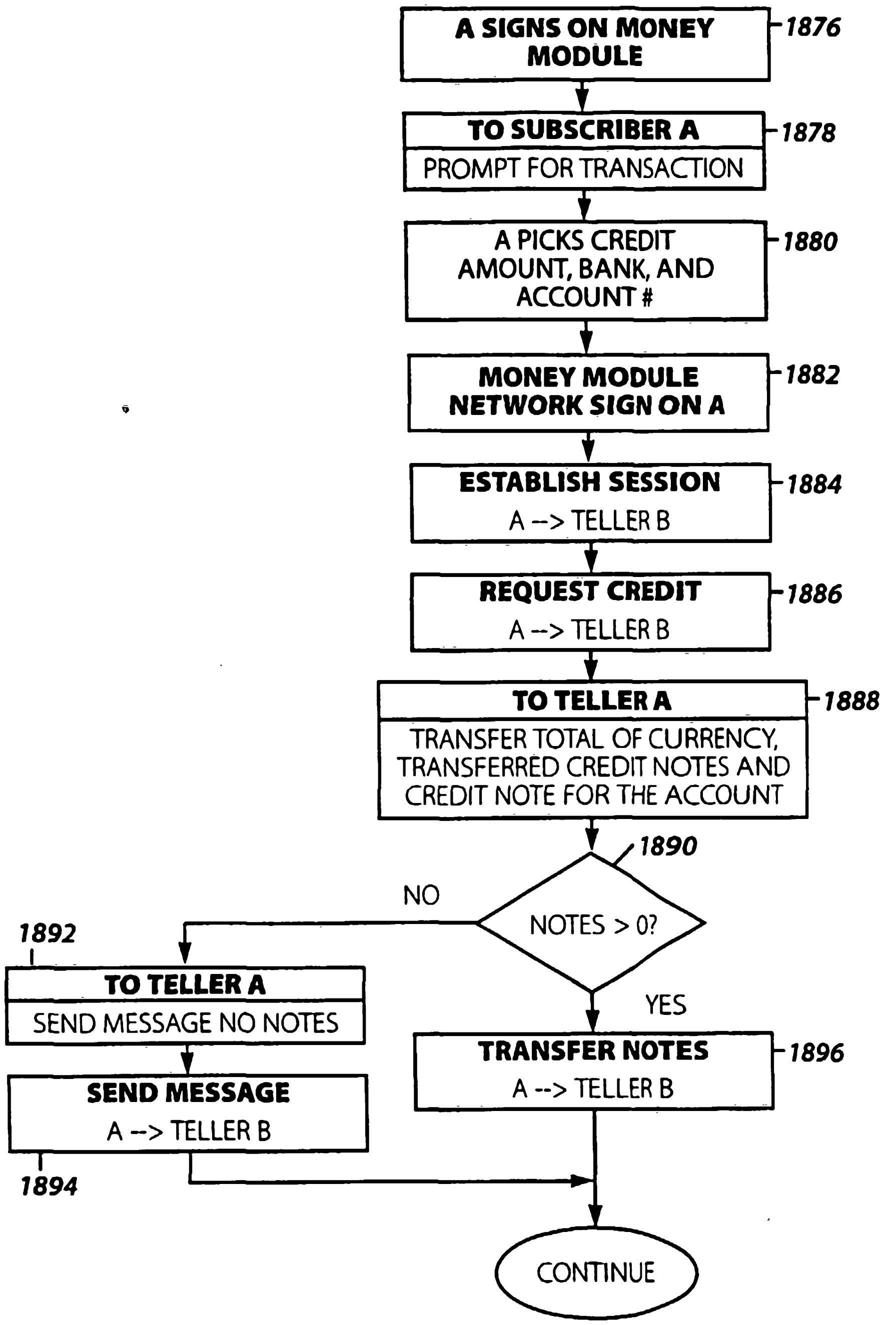

A SIGNS ON MONEY MODULE
1876
TO SUBSCRIBER A
PROMPT FOR TRANSACTION
1878
A PICKS CREDIT AMOUNT, BANK, AND ACCOUNT #
1880
MONEY MODULE NETWORK SIGN ON A
1882
ESTABLISH SESSION
A --> TELLER B
1884
REQUEST CREDIT
A --> TELLER B
1886
TO TELLER A
TRANSFER TOTAL OF CURRENCY, TRANSFERRED CREDIT NOTES AND CREDIT NOTE FOR THE ACCOUNT
1888
1890
NOTES > 0?
NO
YES
1892
TO TELLER A
SEND MESSAGE NO NOTES
SEND MESSAGE
A --> TELLER B
1894
TRANSFER NOTES
A --> TELLER B
1896
CONTINUE

Figure 14

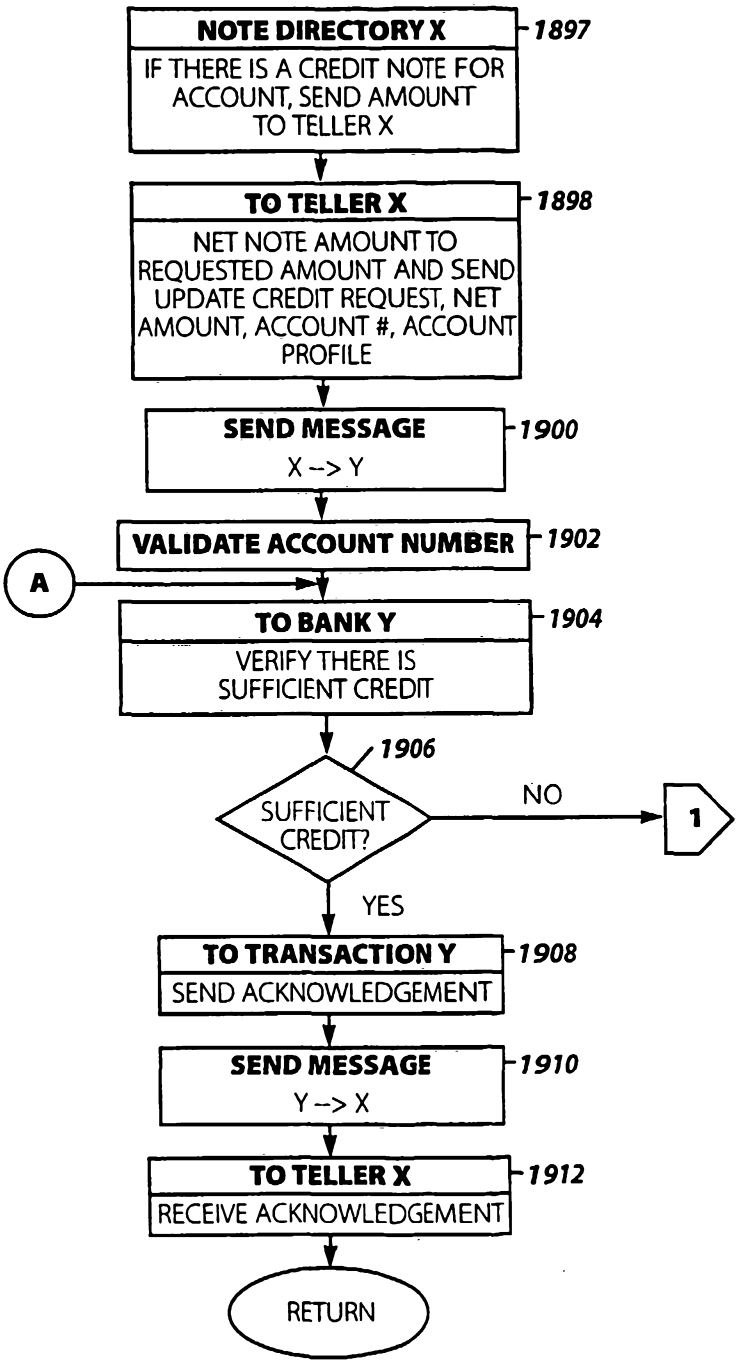

NOTE DIRECTORY X
IF THERE IS A CREDIT NOTE FOR ACCOUNT, SEND AMOUNT TO TELLER X
1897
TO TELLER X
NET NOTE AMOUNT TO REQUESTED AMOUNT AND SEND UPDATE CREDIT REQUEST, NET AMOUNT, ACCOUNT #, ACCOUNT PROFILE
1898
SEND MESSAGE
X --> Y
1900
VALIDATE ACCOUNT NUMBER
1902
A
TO BANK Y
VERIFY THERE IS SUFFICIENT CREDIT
1904
1906
SUFFICIENT CREDIT?
NO
1
YES
TO TRANSACTION Y
SEND ACKNOWLEDGEMENT
1908
SEND MESSAGE
Y --> X
1910
TO TELLER X
RECEIVE ACKNOWLEDGEMENT
1912
RETURN

Figure 15A

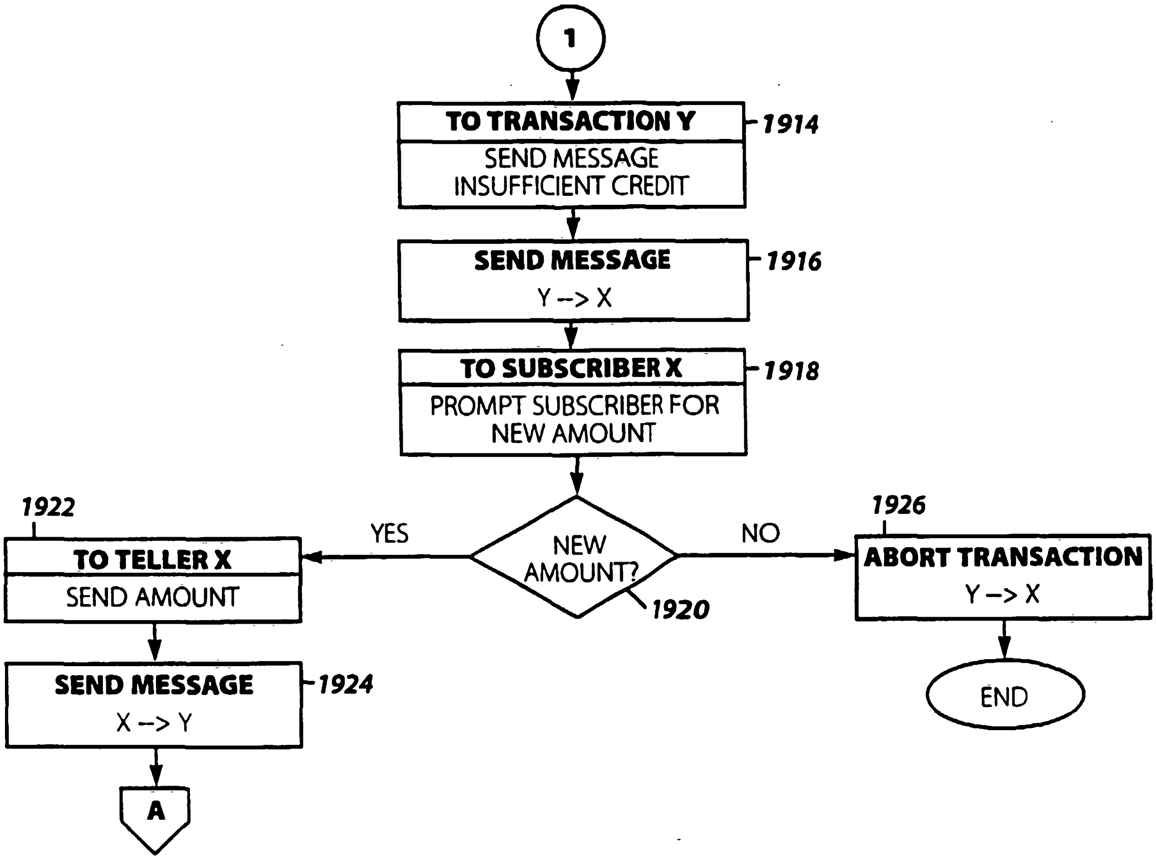

1
TO TRANSACTION Y
SEND MESSAGE INSUFFICIENT CREDIT
1914
SEND MESSAGE
Y –> X
1916
TO SUBSCRIBER X
PROMPT SUBSCRIBER FOR NEW AMOUNT
1918
NEW AMOUNT?
1920
YES
NO
1922
TO TELLER X
SEND AMOUNT
SEND MESSAGE
X –> Y
1924
A
1926
ABORT TRANSACTION
Y –> X
END

Figure 15B

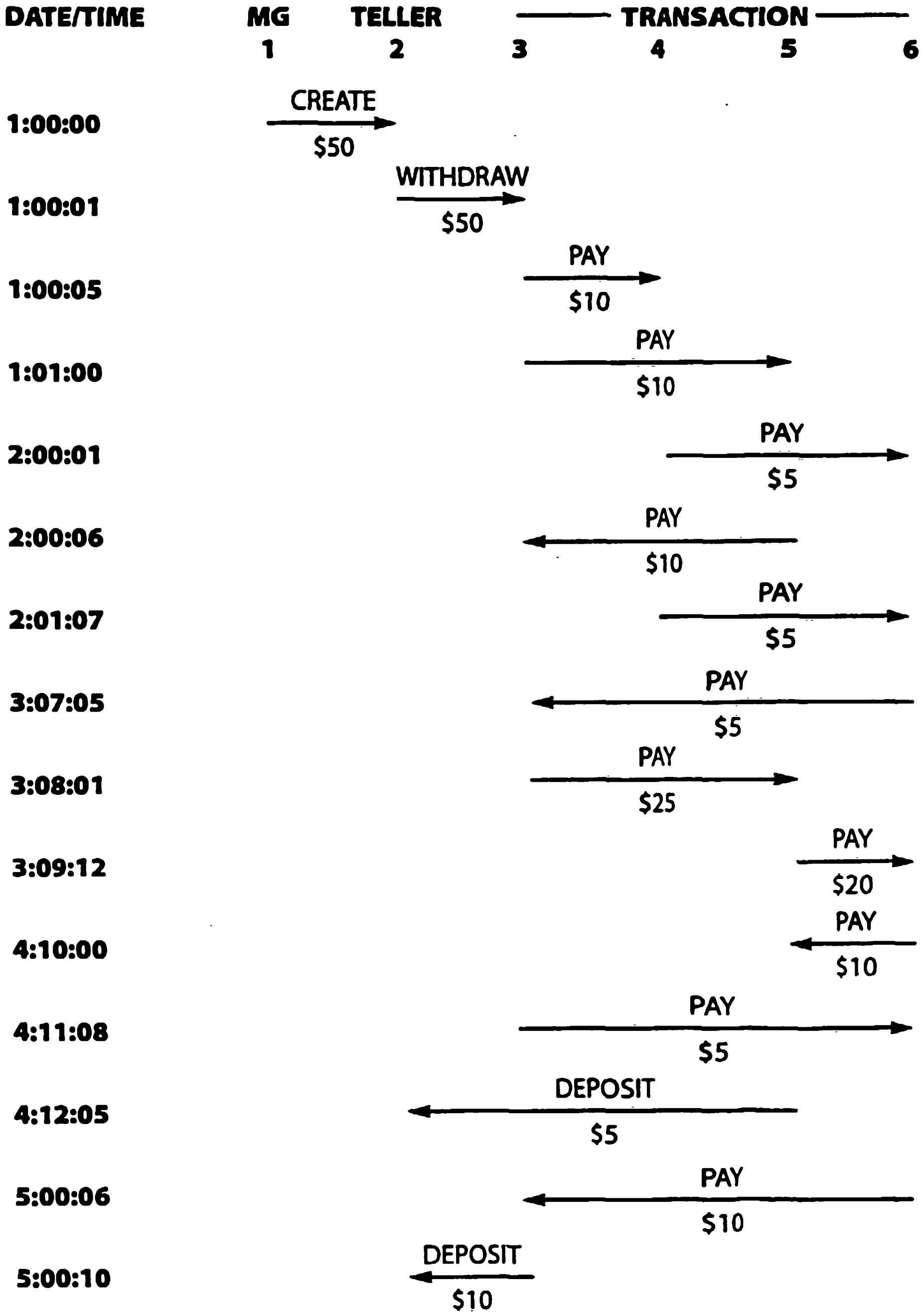
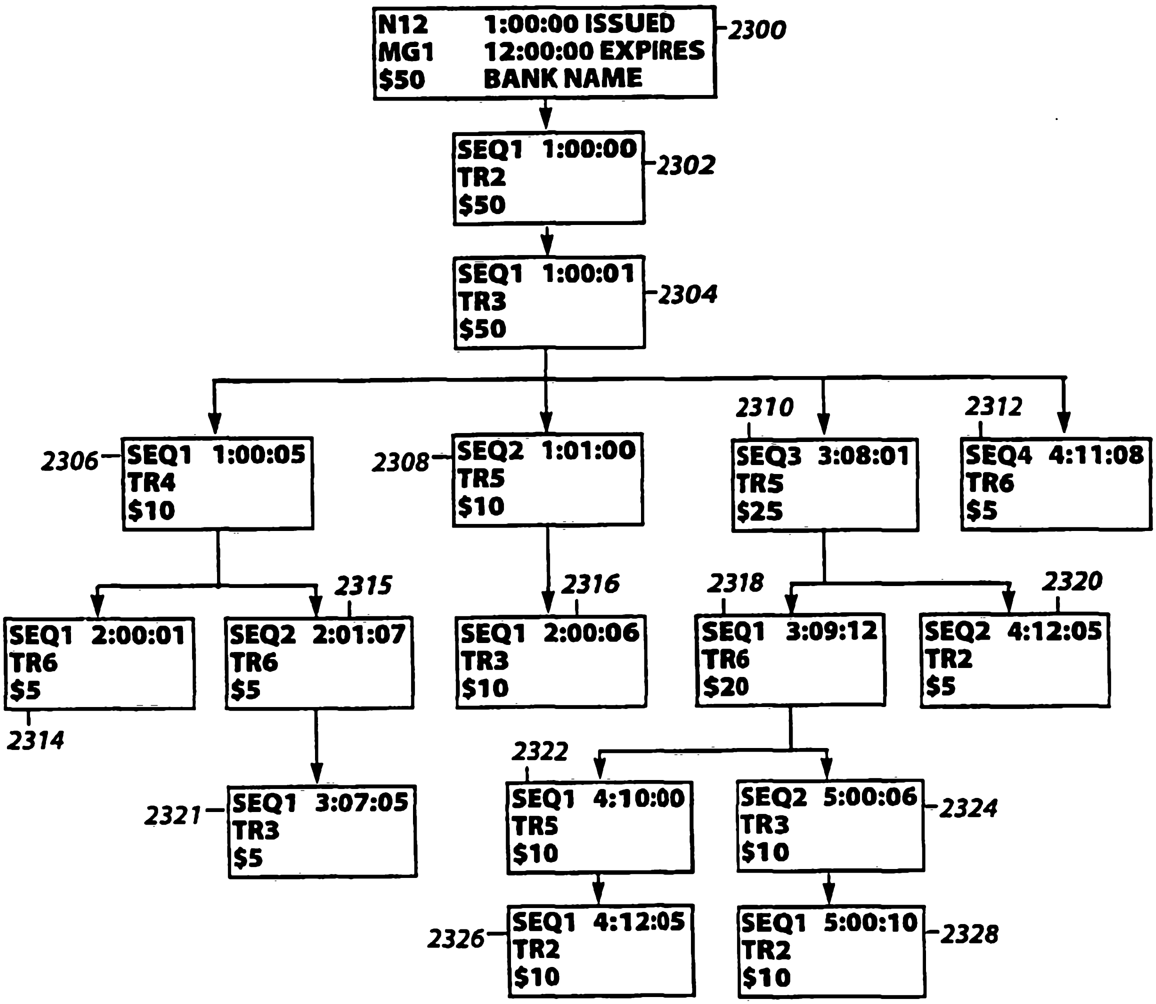

DATE/TIME
MG
TELLER
TRANSACTION
1
2
3
4
5
6
1:00:00
CREATE
$50
1:00:01
WITHDRAW
$50
1:00:05
PAY
$10
1:01:00
PAY
$10
2:00:01
PAY
$5
2:00:06
PAY
$10
2:01:07
PAY
$5
3:07:05
PAY
$5
3:08:01
PAY
$25
3:09:12
PAY
$20
4:10:00
PAY
$10
4:11:08
PAY
$5
4:12:05
DEPOSIT
$5
5:00:06
PAY
$10
5:00:10
DEPOSIT
$10

Figure 16



SEQ = SEQUENCE NUMBER
TR = TRANSFEREE
D:TT:TT = DATE:TIME

Figure 17

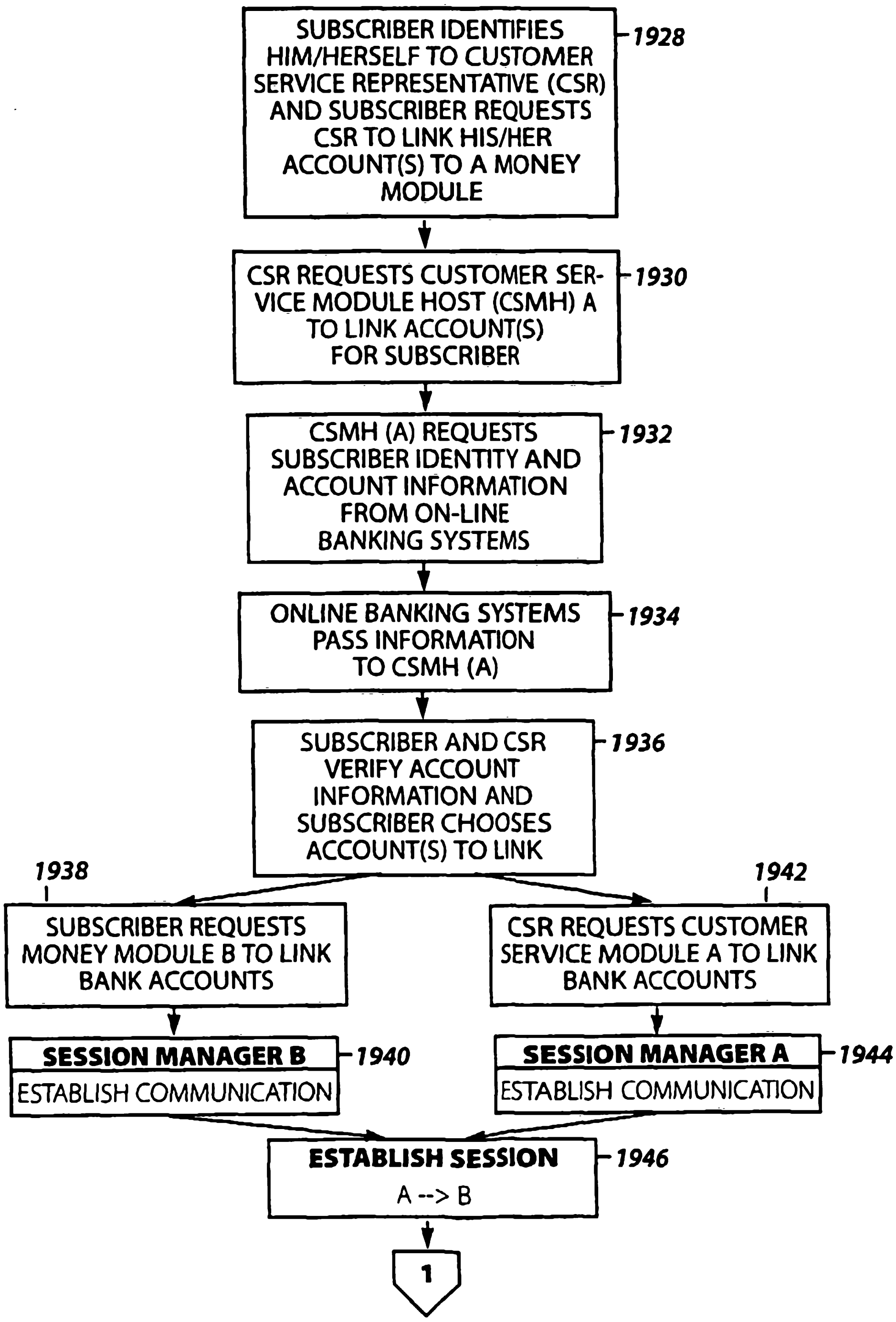

SUBSCRIBER IDENTIFIES HIM/HERSELF TO CUSTOMER SERVICE REPRESENTATIVE (CSR) AND SUBSCRIBER REQUESTS CSR TO LINK HIS/HER ACCOUNT(S) TO A MONEY MODULE
1928
CSR REQUESTS CUSTOMER SERVICE MODULE HOST (CSMH) A TO LINK ACCOUNT(S) FOR SUBSCRIBER
1930
CSMH (A) REQUESTS SUBSCRIBER IDENTITY AND ACCOUNT INFORMATION FROM ON-LINE BANKING SYSTEMS
1932
ONLINE BANKING SYSTEMS PASS INFORMATION TO CSMH (A)
1934
SUBSCRIBER AND CSR VERIFY ACCOUNT INFORMATION AND SUBSCRIBER CHOOSES ACCOUNT(S) TO LINK
1936
1938
SUBSCRIBER REQUESTS MONEY MODULE B TO LINK BANK ACCOUNTS
1942
CSR REQUESTS CUSTOMER SERVICE MODULE A TO LINK BANK ACCOUNTS
SESSION MANAGER B
ESTABLISH COMMUNICATION
1940
SESSION MANAGER A
ESTABLISH COMMUNICATION
1944
ESTABLISH SESSION
A --> B
1946
1

Figure 18A

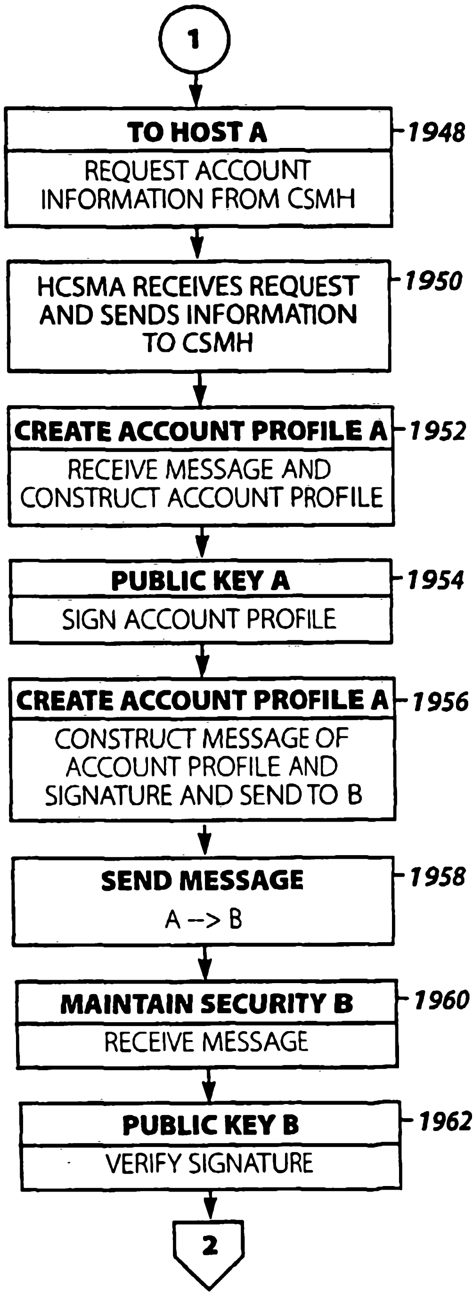

1
TO HOST A
REQUEST ACCOUNT INFORMATION FROM CSMH
1948
HCSMA RECEIVES REQUEST AND SENDS INFORMATION TO CSMH
1950
CREATE ACCOUNT PROFILE A
RECEIVE MESSAGE AND CONSTRUCT ACCOUNT PROFILE
1952
PUBLIC KEY A
SIGN ACCOUNT PROFILE
1954
CREATE ACCOUNT PROFILE A
CONSTRUCT MESSAGE OF ACCOUNT PROFILE AND SIGNATURE AND SEND TO B
1956
SEND MESSAGE
A --> B
1958
MAINTAIN SECURITY B
RECEIVE MESSAGE
1960
PUBLIC KEY B
VERIFY SIGNATURE
1962
2

Figure 18B

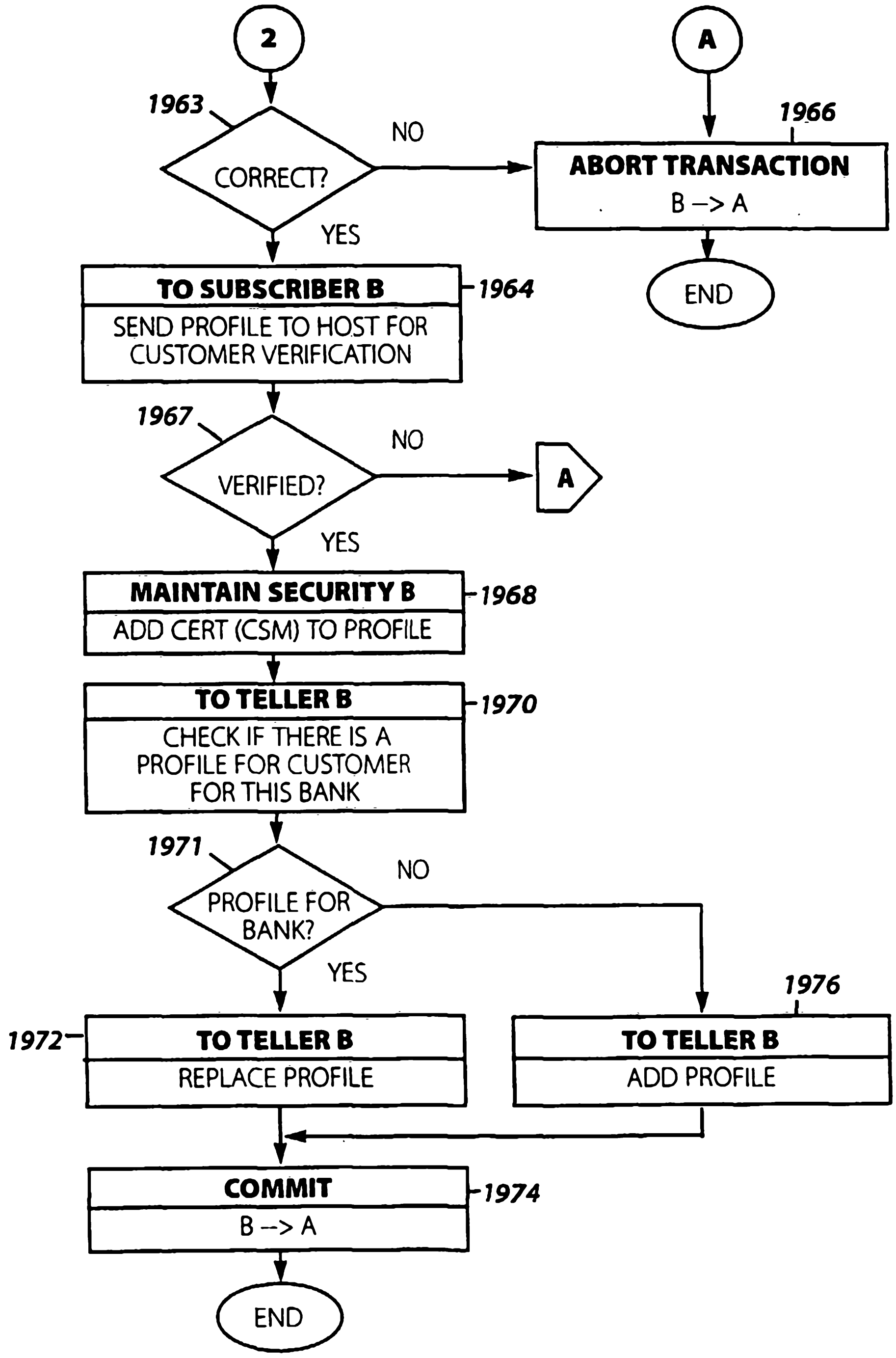

2
A
1963
CORRECT?
NO
YES
1966
ABORT TRANSACTION
B -> A
END
TO SUBSCRIBER B
SEND PROFILE TO HOST FOR CUSTOMER VERIFICATION
1964
1967
VERIFIED?
NO
A
YES
MAINTAIN SECURITY B
ADD CERT (CSM) TO PROFILE
1968
TO TELLER B
CHECK IF THERE IS A PROFILE FOR CUSTOMER FOR THIS BANK
1970
1971
PROFILE FOR BANK?
NO
YES
1976
1972
TO TELLER B
REPLACE PROFILE
TO TELLER B
ADD PROFILE
COMMIT
B -> A
1974
END

Figure 18C

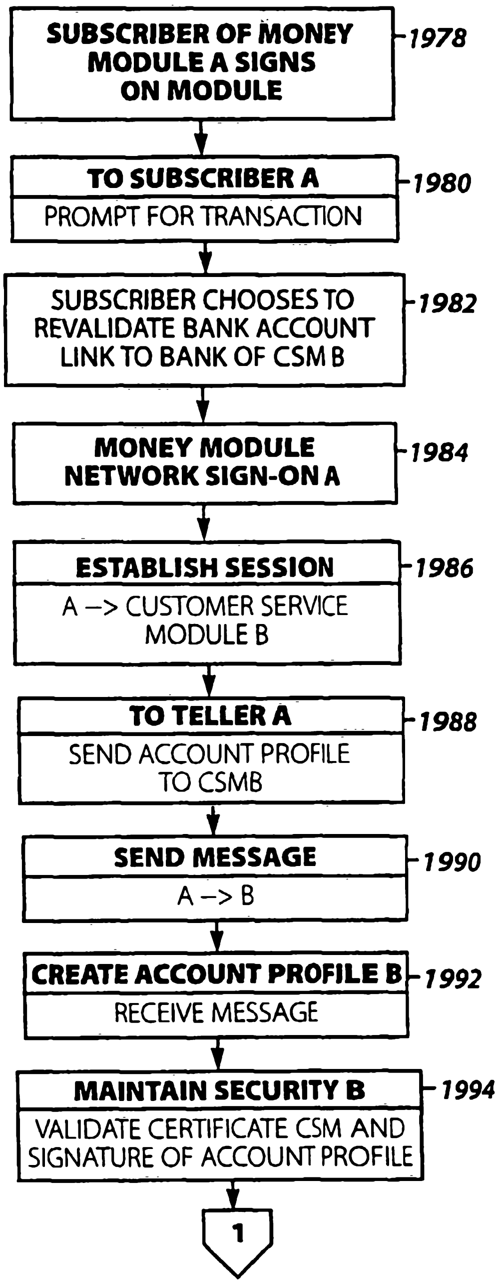

SUBSCRIBER OF MONEY MODULE A SIGNS ON MODULE
1978
TO SUBSCRIBER A
PROMPT FOR TRANSACTION
1980
SUBSCRIBER CHOOSES TO REVALIDATE BANK ACCOUNT LINK TO BANK OF CSM B
1982
MONEY MODULE NETWORK SIGN-ON A
1984
ESTABLISH SESSION
A –> CUSTOMER SERVICE MODULE B
1986
TO TELLER A
SEND ACCOUNT PROFILE TO CSMB
1988
SEND MESSAGE
A –> B
1990
CREATE ACCOUNT PROFILE B
RECEIVE MESSAGE
1992
MAINTAIN SECURITY B
VALIDATE CERTIFICATE CSM AND SIGNATURE OF ACCOUNT PROFILE
1994
1

Figure 19A

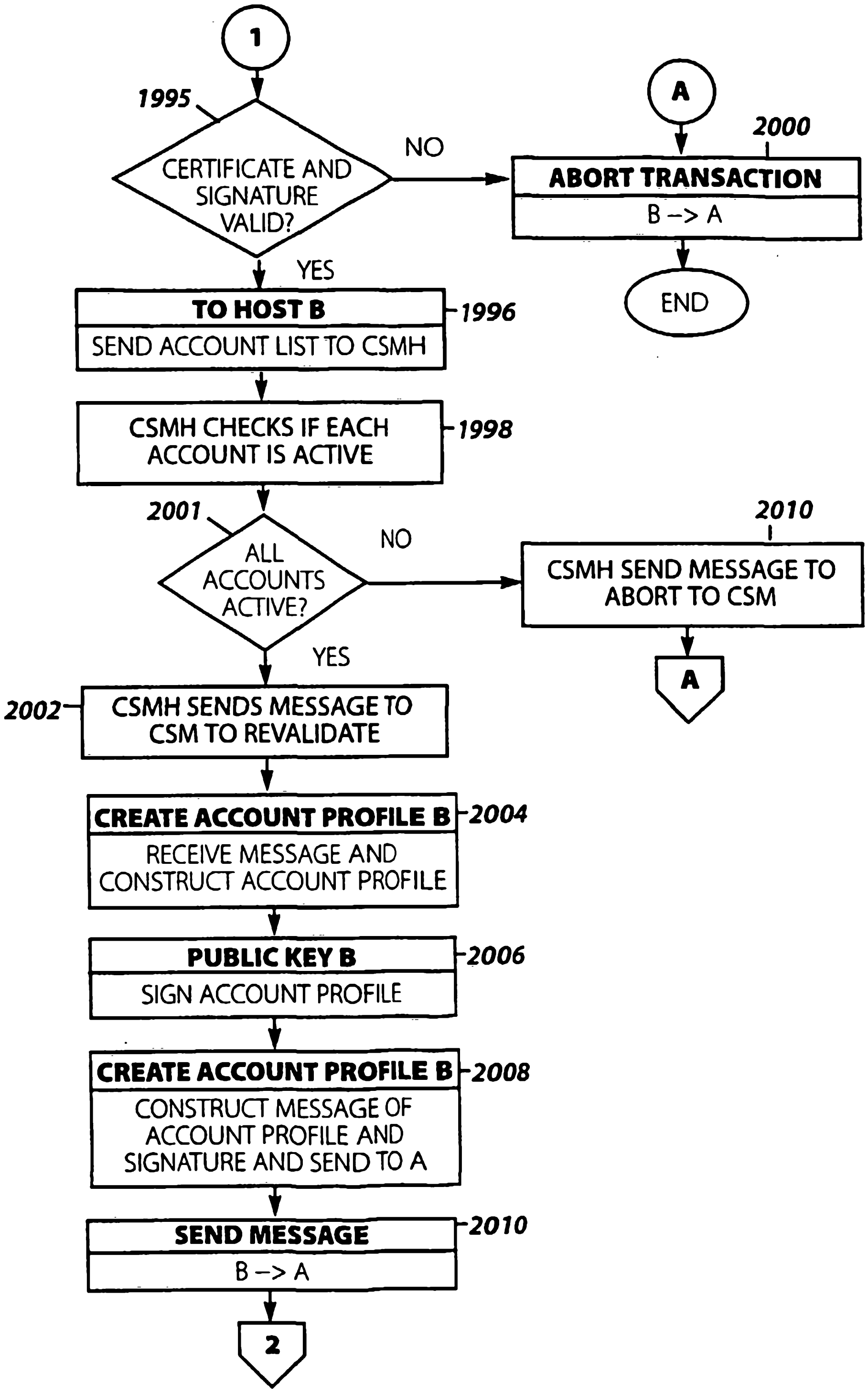

1
A
1995
CERTIFICATE AND SIGNATURE VALID?
NO
2000
ABORT TRANSACTION
B -> A
END
YES
TO HOST B
SEND ACCOUNT LIST TO CSMH
1996
CSMH CHECKS IF EACH ACCOUNT IS ACTIVE
1998
2001
ALL ACCOUNTS ACTIVE?
NO
2010
CSMH SEND MESSAGE TO ABORT TO CSM
A
YES
2002
CSMH SENDS MESSAGE TO CSM TO REVALIDATE
CREATE ACCOUNT PROFILE B
RECEIVE MESSAGE AND CONSTRUCT ACCOUNT PROFILE
2004
PUBLIC KEY B
SIGN ACCOUNT PROFILE
2006
CREATE ACCOUNT PROFILE B
CONSTRUCT MESSAGE OF ACCOUNT PROFILE AND SIGNATURE AND SEND TO A
2008
SEND MESSAGE
B -> A
2010
2

Figure 19B

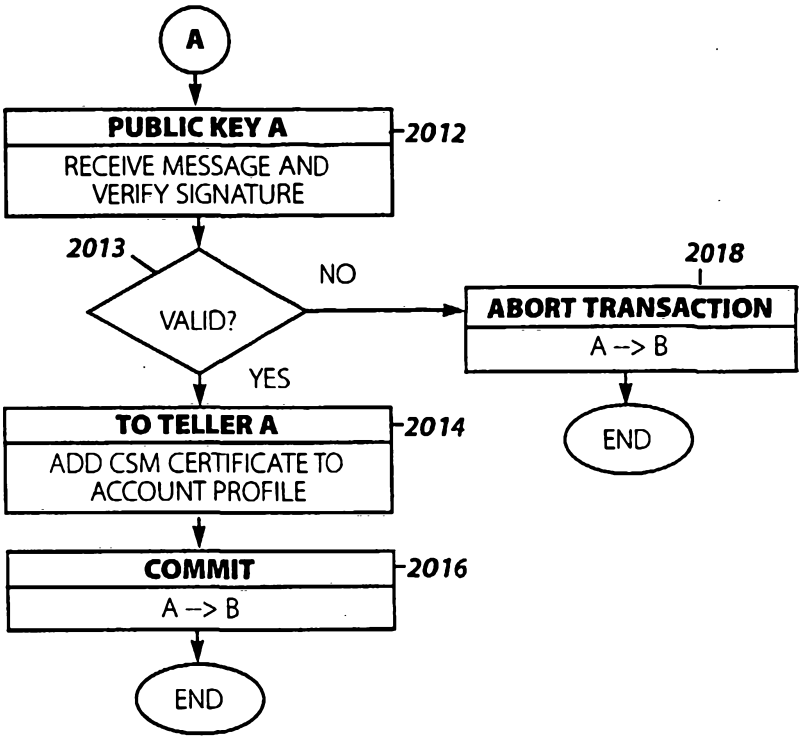
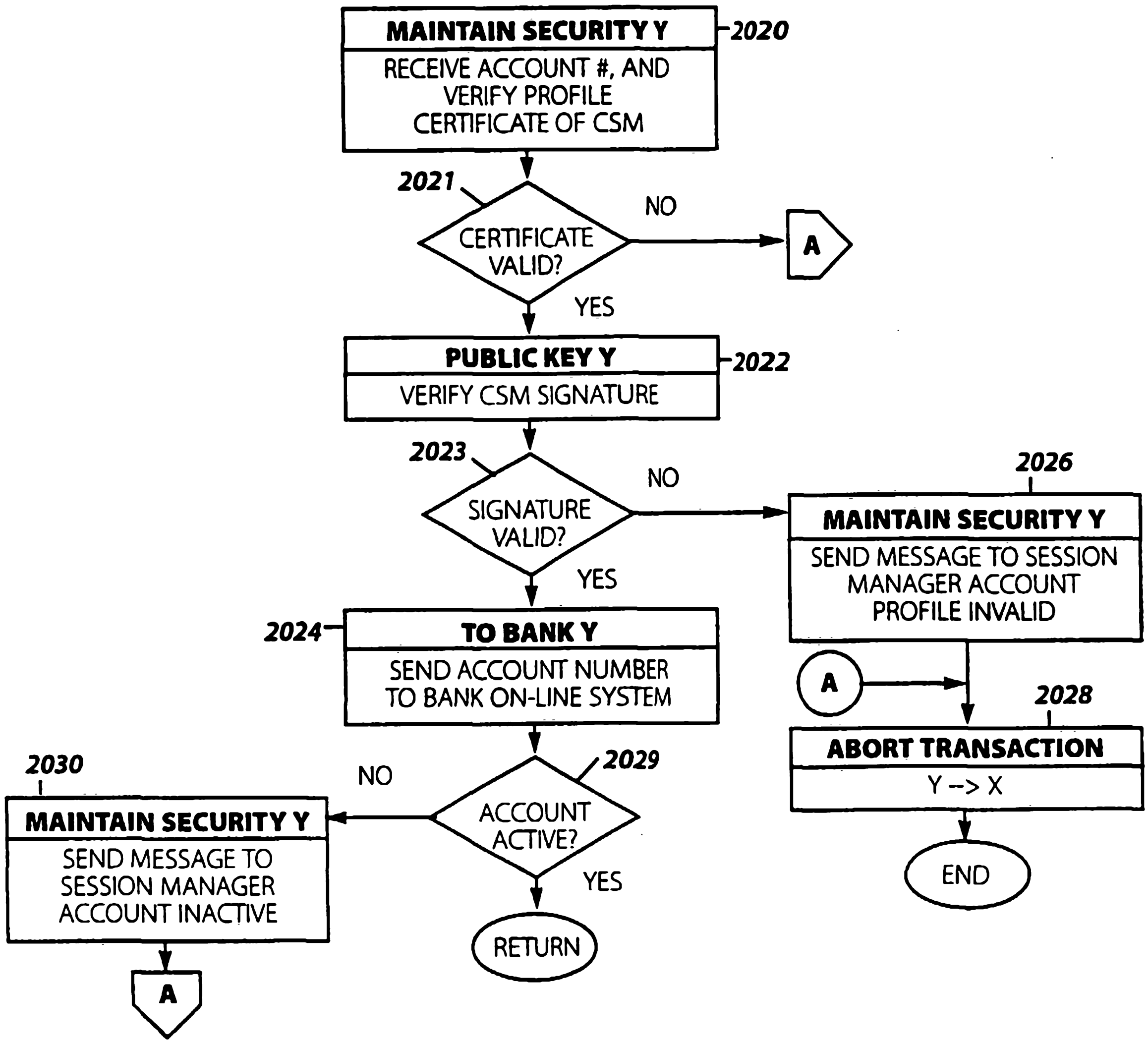

A
PUBLIC KEY A
RECEIVE MESSAGE AND VERIFY SIGNATURE
2012
2013
VALID?
NO
YES
2018
ABORT TRANSACTION
A --> B
END
TO TELLER A
ADD CSM CERTIFICATE TO ACCOUNT PROFILE
2014
COMMIT
A --> B
2016
END

Figure 19C



MAINTAIN SECURITY Y
RECEIVE ACCOUNT #, AND VERIFY PROFILE CERTIFICATE OF CSM
2020
2021
CERTIFICATE VALID?
NO
YES
A
PUBLIC KEY Y
VERIFY CSM SIGNATURE
2022
2023
SIGNATURE VALID?
NO
YES
2026
MAINTAIN SECURITY Y
SEND MESSAGE TO SESSION MANAGER ACCOUNT PROFILE INVALID
2024
TO BANK Y
SEND ACCOUNT NUMBER TO BANK ON-LINE SYSTEM
A
2028
ABORT TRANSACTION
Y --> X
END
2030
MAINTAIN SECURITY Y
SEND MESSAGE TO SESSION MANAGER ACCOUNT INACTIVE
A
NO
2029
ACCOUNT ACTIVE?
YES
RETURN

Figure 20

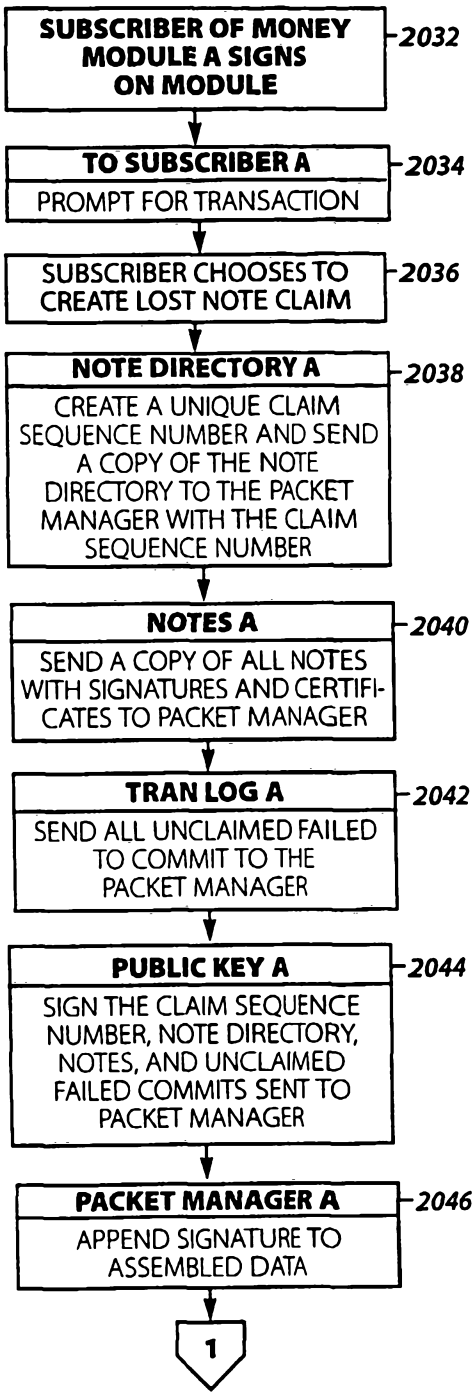

SUBSCRIBER OF MONEY MODULE A SIGNS ON MODULE
2032
TO SUBSCRIBER A
PROMPT FOR TRANSACTION
2034
SUBSCRIBER CHOOSES TO CREATE LOST NOTE CLAIM
2036
NOTE DIRECTORY A
CREATE A UNIQUE CLAIM SEQUENCE NUMBER AND SEND A COPY OF THE NOTE DIRECTORY TO THE PACKET MANAGER WITH THE CLAIM SEQUENCE NUMBER
2038
NOTES A
SEND A COPY OF ALL NOTES WITH SIGNATURES AND CERTIFICATES TO PACKET MANAGER
2040
TRAN LOG A
SEND ALL UNCLAIMED FAILED TO COMMIT TO THE PACKET MANAGER
2042
PUBLIC KEY A
SIGN THE CLAIM SEQUENCE NUMBER, NOTE DIRECTORY, NOTES, AND UNCLAIMED FAILED COMMITS SENT TO PACKET MANAGER
2044
PACKET MANAGER A
APPEND SIGNATURE TO ASSEMBLED DATA
2046
1

Figure 21A

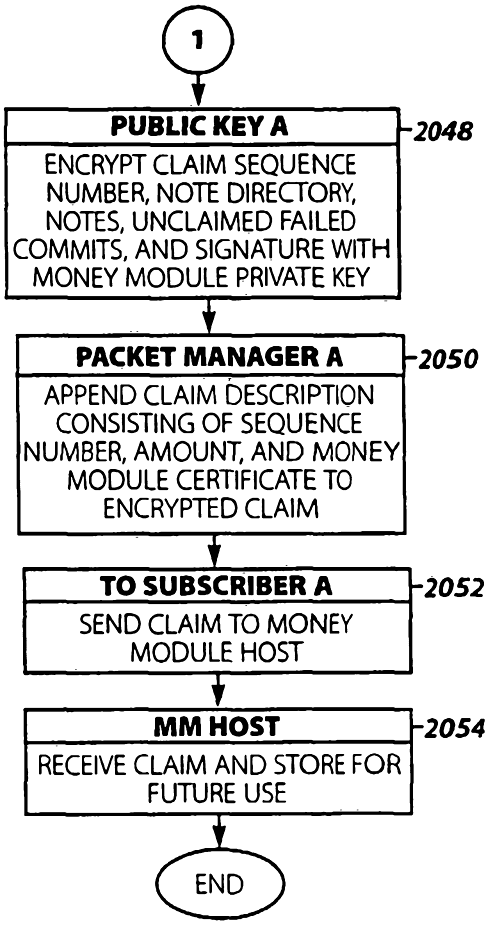

1
PUBLIC KEY A
ENCRYPT CLAIM SEQUENCE NUMBER, NOTE DIRECTORY, NOTES, UNCLAIMED FAILED COMMITS, AND SIGNATURE WITH MONEY MODULE PRIVATE KEY
2048
PACKET MANAGER A
APPEND CLAIM DESCRIPTION CONSISTING OF SEQUENCE NUMBER, AMOUNT, AND MONEY MODULE CERTIFICATE TO ENCRYPTED CLAIM
2050
TO SUBSCRIBER A
SEND CLAIM TO MONEY MODULE HOST
2052
MM HOST
RECEIVE CLAIM AND STORE FOR FUTURE USE
2054
END

Figure 21B

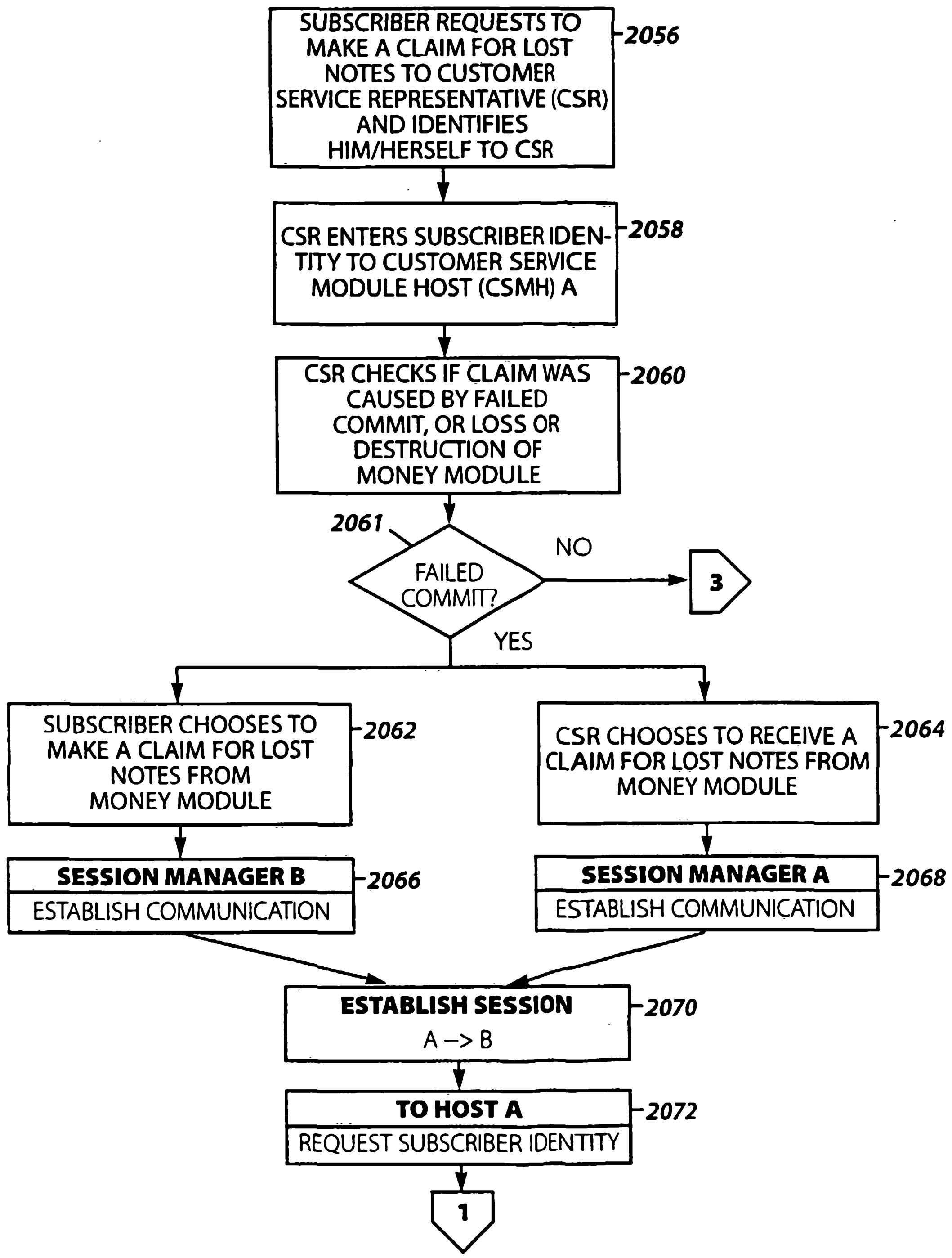

SUBSCRIBER REQUESTS TO MAKE A CLAIM FOR LOST NOTES TO CUSTOMER SERVICE REPRESENTATIVE (CSR) AND IDENTIFIES HIM/HERSELF TO CSR
2056
CSR ENTERS SUBSCRIBER IDENTITY TO CUSTOMER SERVICE MODULE HOST (CSMH) A
2058
CSR CHECKS IF CLAIM WAS CAUSED BY FAILED COMMIT, OR LOSS OR DESTRUCTION OF MONEY MODULE
2060
2061
FAILED COMMIT?
NO
3
YES
SUBSCRIBER CHOOSES TO MAKE A CLAIM FOR LOST NOTES FROM MONEY MODULE
2062
CSR CHOOSES TO RECEIVE A CLAIM FOR LOST NOTES FROM MONEY MODULE
2064
SESSION MANAGER B
ESTABLISH COMMUNICATION
2066
SESSION MANAGER A
ESTABLISH COMMUNICATION
2068
ESTABLISH SESSION
A -> B
2070
TO HOST A
REQUEST SUBSCRIBER IDENTITY
2072
1

Figure 22A

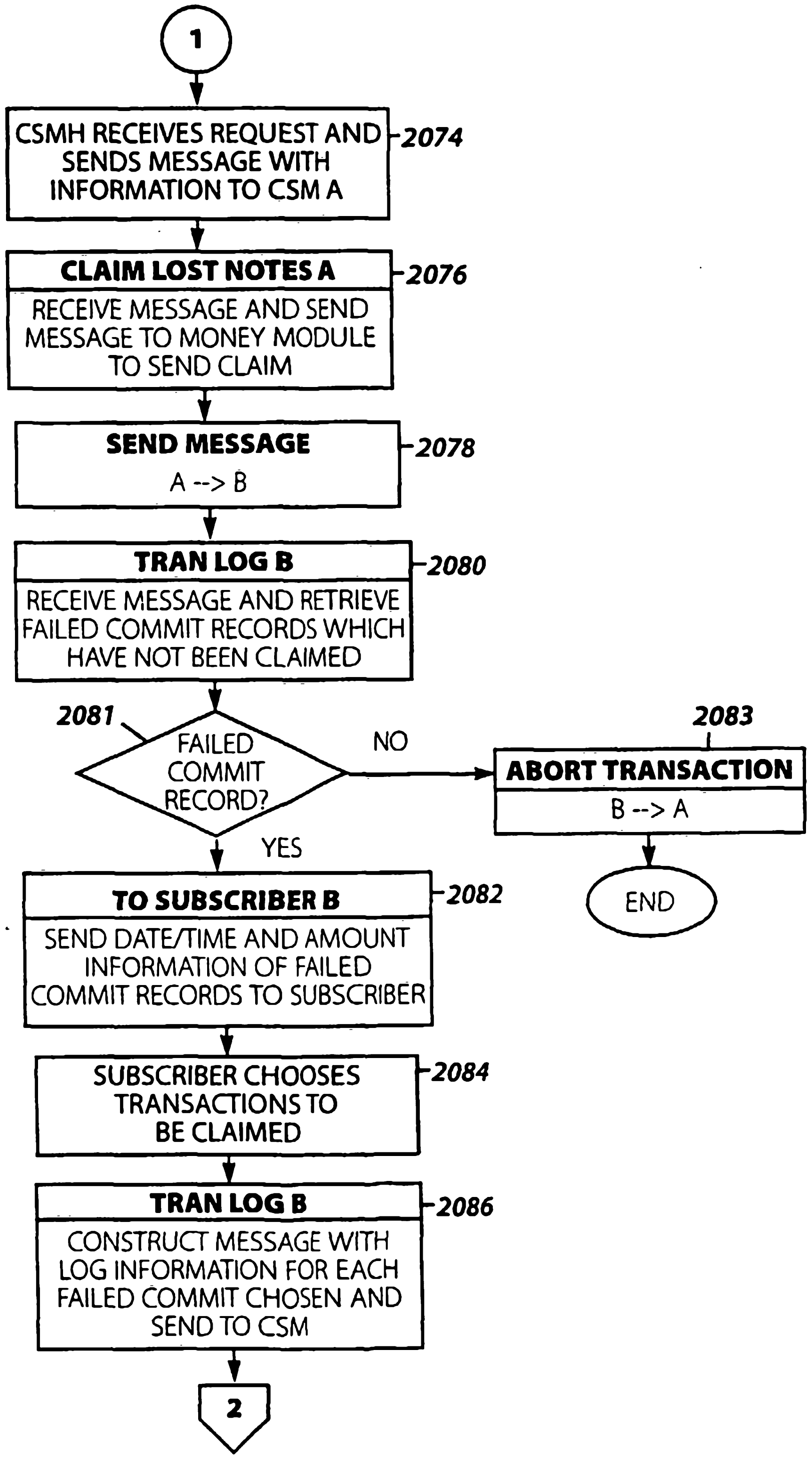

1
CSMH RECEIVES REQUEST AND SENDS MESSAGE WITH INFORMATION TO CSM A
2074
CLAIM LOST NOTES A
RECEIVE MESSAGE AND SEND MESSAGE TO MONEY MODULE TO SEND CLAIM
2076
SEND MESSAGE
A --> B
2078
TRAN LOG B
RECEIVE MESSAGE AND RETRIEVE FAILED COMMIT RECORDS WHICH HAVE NOT BEEN CLAIMED
2080
2081
FAILED COMMIT RECORD?
NO
YES
2083
ABORT TRANSACTION
B --> A
END
TO SUBSCRIBER B
SEND DATE/TIME AND AMOUNT INFORMATION OF FAILED COMMIT RECORDS TO SUBSCRIBER
2082
SUBSCRIBER CHOOSES TRANSACTIONS TO BE CLAIMED
2084
TRAN LOG B
CONSTRUCT MESSAGE WITH LOG INFORMATION FOR EACH FAILED COMMIT CHOSEN AND SEND TO CSM
2086
2

Figure 22B

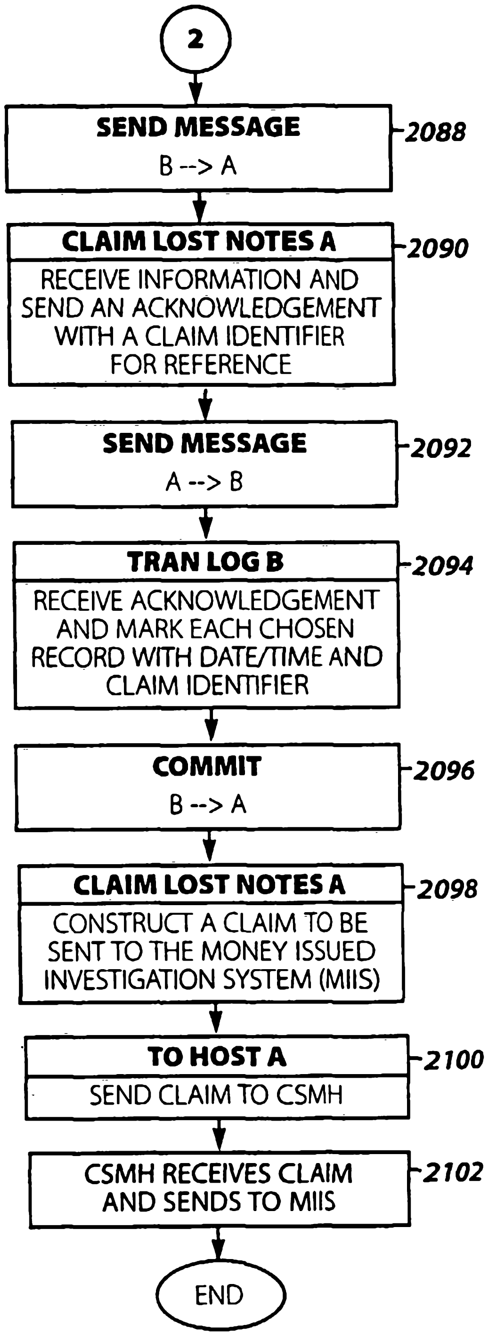

2
SEND MESSAGE
B --> A
2088
CLAIM LOST NOTES A
RECEIVE INFORMATION AND SEND AN ACKNOWLEDGEMENT WITH A CLAIM IDENTIFIER FOR REFERENCE
2090
SEND MESSAGE
A --> B
2092
TRAN LOG B
RECEIVE ACKNOWLEDGEMENT AND MARK EACH CHOSEN RECORD WITH DATE/TIME AND CLAIM IDENTIFIER
2094
COMMIT
B --> A
2096
CLAIM LOST NOTES A
CONSTRUCT A CLAIM TO BE SENT TO THE MONEY ISSUED INVESTIGATION SYSTEM (MIIS)
2098
TO HOST A
SEND CLAIM TO CSMH
2100
CSMH RECEIVES CLAIM AND SENDS TO MIIS
2102
END

Figure 22C

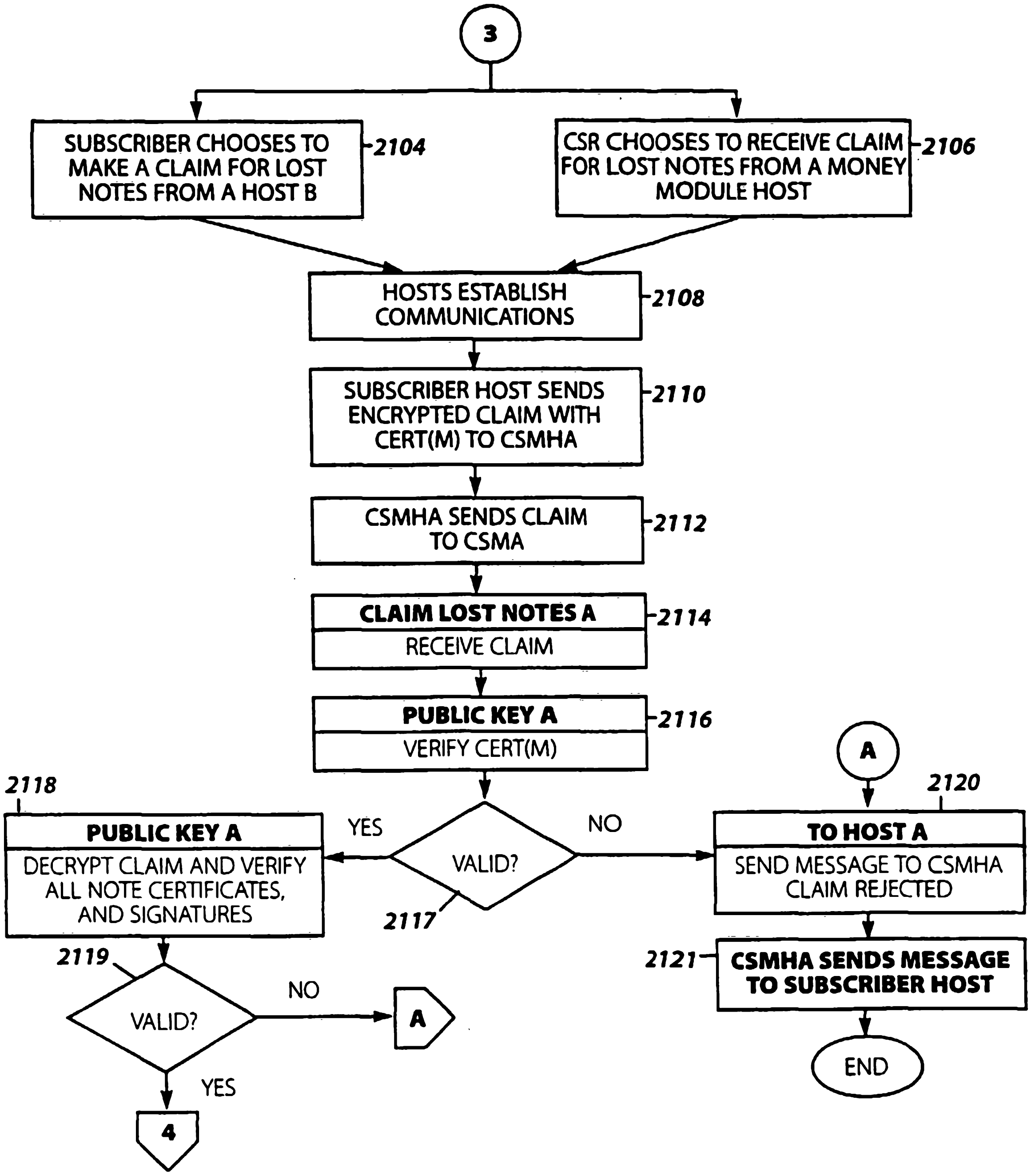

3
SUBSCRIBER CHOOSES TO MAKE A CLAIM FOR LOST NOTES FROM A HOST B
2104
CSR CHOOSES TO RECEIVE CLAIM FOR LOST NOTES FROM A MONEY MODULE HOST
2106
HOSTS ESTABLISH COMMUNICATIONS
2108
SUBSCRIBER HOST SENDS ENCRYPTED CLAIM WITH CERT(M) TO CSMHA
2110
CSMHA SENDS CLAIM TO CSMA
2112
CLAIM LOST NOTES A
RECEIVE CLAIM
2114
PUBLIC KEY A
VERIFY CERT(M)
2116
VALID?
2117
YES
NO
2118
PUBLIC KEY A
DECRYPT CLAIM AND VERIFY ALL NOTE CERTIFICATES, AND SIGNATURES
2119
VALID?
NO
A
YES
4
A
2120
TO HOST A
SEND MESSAGE TO CSMHA CLAIM REJECTED
2121
CSMHA SENDS MESSAGE TO SUBSCRIBER HOST
END

Figure 22D

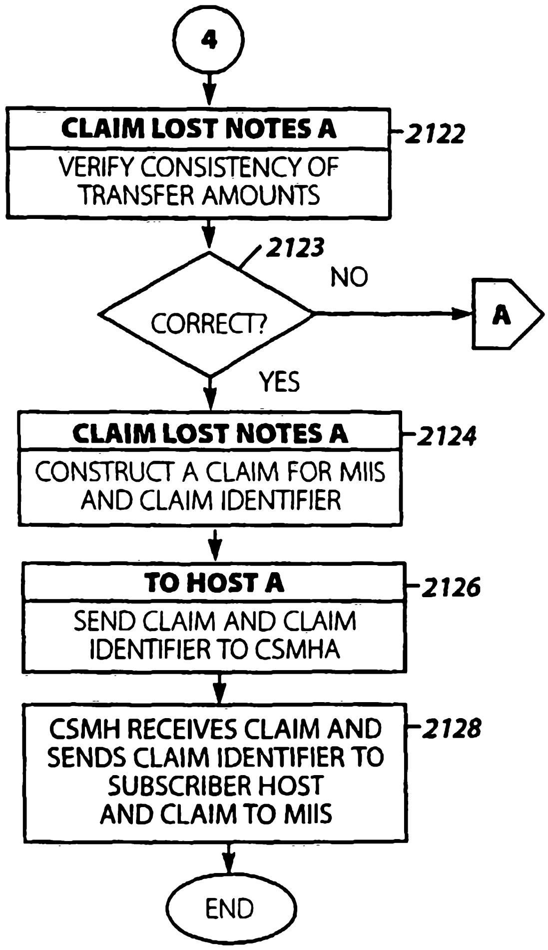

4
CLAIM LOST NOTES A
VERIFY CONSISTENCY OF TRANSFER AMOUNTS
2122
2123
NO
CORRECT?
A
YES
CLAIM LOST NOTES A
CONSTRUCT A CLAIM FOR MIIS AND CLAIM IDENTIFIER
2124
TO HOST A
SEND CLAIM AND CLAIM IDENTIFIER TO CSMHA
2126
CSMH RECEIVES CLAIM AND SENDS CLAIM IDENTIFIER TO SUBSCRIBER HOST AND CLAIM TO MIIS
2128
END

Figure 22E