

**Informed Control Inc. Technical Information Disclosure**  
**"System for authentication with an electronic mail service"**  
**(200612-03)**

**Published: January 8, 2008**

**BACKGROUND**

This disclosure relates generally to computer security, in particular to the use of authentication for security of access to an electronic mail server on a computer network.

**DRAWINGS -- Figures**

FIG. 1 is a diagram that illustrates the components of the system for authentication with an electronic mail service.

FIG. 2A, FIG. 2B, FIG. 2C, FIG. 2D, and FIG. 2E are a flowchart illustrating the operation of the authentication process in a client mail application.

FIG. 3 is a flowchart illustrating the operation of a listening thread in a mail server.

FIG. 4A, FIG. 4B, FIG. 4C, FIG. 4D and FIG. 4E are a flowchart illustrating the operation of a connection thread in a mail server.

FIG. 5 is a flowchart illustrating the operation of a mail system Security Token Service (STS) STS server.

## **“System for authentication in an electronic mail service” continued**

FIG. 6 is an example of a protocol exchange of the negotiation of the CARD-INLINE SASL mechanism in a Simple Mail Transfer Protocol (SMTP) connection.

FIG. 7 is an example of a protocol exchange of the negotiation of the CARD-INLINE SASL mechanism in an Internet Message Access Protocol (IMAP) connection.

FIG. 8 is an example of a protocol exchange of the negotiation of the CARD-INLINE SASL mechanism in a Post Office Protocol (POP) connection.

FIG. 9 is an example of a protocol exchange of the negotiation of the CARD-RPSTS SASL mechanism in a Simple Mail Transfer Protocol (SMTP) connection.

FIG. 10 is an example of a protocol exchange of the negotiation of the CARD-RPSTS SASL mechanism in an Internet Message Access Protocol (IMAP) connection.

FIG. 11 is an example of a protocol exchange of the negotiation of the CARD-RPSTS SASL mechanism in a Post Office Protocol (POP) connection.

FIG. 12 is an example of the components of a computer network which provides an electronic mail service.

### **DESCRIPTION - One Embodiment**

An Identity Metasystem is a collection of interoperable computing elements on a computer network which enables users of the services provided by the network to manage and exchange their digital identities. In an Identity Metasystem, an Identity Provider is a network server responsible for authenticating users, and a Relying Party is a network server which requires an authenticated user identity in order to provide service. The Identity Metasystem

## **“System for authentication in an electronic mail service” continued**

defines the mechanisms that enable a Relying Party to validate that a user requesting service from that Relying Party has been previously authenticated by an Identity Provider, in which the Relying Party is a web service based on the Simple Object Access Protocol (SOAP), or web server based on the Hypertext Transfer Protocol (HTTP).

The document "A Technical Reference for InfoCard v1.0 in Windows", published in August 2005 by Microsoft Corporation, describes the network communication protocols by which an Identity Selector may obtain the token requirements of a Relying Party, then authenticate to an Identity Provider, and finally send a token obtained from an Identity Provider to a Relying Party. The protocols defined in "A Technical Reference for InfoCard v1.0 in Windows" specify a protocol exchange in which the protocols defined in the documents Web Services Security: SOAP Message Security 1.0 (WS-Security 2004), Web Services Trust Language (WS-Trust), Web Services Security Policy Language (WS-SecurityPolicy) and Web Services Metadata Exchange (WS-MetadataExchange), all of which are based on the Simple Object Access Protocol (SOAP), are to be used for the communication between the Identity Selector and the Relying Party. The Simple Object Access Protocol is typically used only between applications in a web services framework.

The document "A Guide to Supporting InfoCard v1.0 Within Web Applications and Browsers", published in March 2006 by Microsoft Corporation, describes the network communication protocols by which an Identity Selector may obtain the token requirements of a Relying Party and send a token obtained from an Identity Provider to a Relying Party using the Hypertext Transfer Protocol (HTTP) and Hypertext Markup Language (HTML). The Hypertext

## **“System for authentication in an electronic mail service” continued**

Transfer Protocol is typically used by a web browser to communicate with a web server to web application.

This disclosure describes how an electronic mail client can demonstrate to an electronic mail server that the user operating the electronic mail client has been authenticated.

An electronic mail system is a collection of one or more mail servers and one or more mail clients. Mail clients submit new messages created by users to mail servers using the Simple Mail Transfer Protocol (SMTP). Mail clients retrieve the user's messages using either the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP). Examples of mail clients include Microsoft Outlook Express, Netscape Communicator, Mozilla Thunderbird as well as specialized applications running in cellular phones and other mobile devices.

A mail server requires an authenticated client identity in order to retrieve messages, to ensure that a client only retrieves message on behalf of the user operating that client. A mail server may also require an authenticated client identity in order to submit new messages, in order to prevent unauthorized users from using the mail system's resources to distribute mail.

The specifications in the two above-mentioned documents "A Technical Reference for InfoCard v1.0 in Windows" and "A Guide to Supporting InfoCard v1.0 Within Web Applications and Browsers" are by themselves not adequate to enable an electronic mail system to be part of an Identity Metasystem, as communication between an electronic mail client and an electronic mail server typically occurs using the IMAP, POP and SMTP protocols, which are neither part of a web services framework, nor based on the Hypertext Transfer Protocol.

## **“System for authentication in an electronic mail service” continued**

For authentication, the electronic mail protocols IMAP, POP and SMTP leverage the security framework of the Simple Authentication and Security Layer (SASL). The SASL framework is described in the document "Simple Authentication and Security Layer (SASL)" by Alexey Melnikov and Kurt Zeilenga, published in 2006. In the IMAP, POP and SMTP protocols use of SASL, the mail client requests from the mail server the use of a particular SASL mechanism, the mail server replies with an initial challenge, the mail client then sends to the mail server an initial response, and finally the mail server replies with the outcome of the authentication. Existing SASL mechanisms, however, do not support the Identity Metasystem SOAP or HTTP-based protocols for sending the outcome of authentication to a mail server as the Relying Party.

This disclosure defines a new SASL mechanism, CARD-INLINE. The name of the CARD-INLINE SASL mechanism is "CARD-INLINE". The mechanism is server first, does not transfer an authorization identity string, does not offer a security layer, and requires the use of an underlying integrity layer, Transport Layer Security (TLS). This SASL mechanism consists of two stages of a protocol exchange that occur in the electronic mail protocol: the mail server sends a server-challenge to the mail client, and the mail client replies to the mail server with a client-response. In this SASL mechanism, the mail server does not provide additional data when indicating a successful outcome.

In the CARD-INLINE mechanism, the server-challenge is a protocol data unit (PDU) sent from the mail server to the mail client. The server-challenge PDU consists of a base64 encoding of a XML-encoded document, specifying the mail server's requirements for client authentication. The format of the document is defined by the Web Services Security Policy Language, and the

## **“System for authentication in an electronic mail service” continued**

representation of the mail server's policy is defined by section 4.1 of "A Technical Reference for InfoCard v1.0 in Windows".

The client-response in the CARD-INLINE mechanism is a protocol data unit sent from the mail client to the mail server. The client-response consists of a base64 encoding of an XML-encoded document, representing an encrypted authentication token from the Identity Provider. The token is a Security Assertion Markup Language (SAML) assertion, in a format defined in the document "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", edited by Scott Cantor, John Kemp, Rob Philpott and Eve Maler. The token is encrypted by a randomly generated symmetric key, and this key is encrypted with the public key of the mail server, obtained from the TLS server certificate sent to the mail client by the mail server. The encryption is performed according to the specification contained in the document "XML Encryption Syntax and Processing" by Donald Eastlake, Joseph Reagle, Takeshi Imamura, Blair Dillaway, and Ed Simon.

An illustration of the CARD-INLINE SASL exchange between a mail client (indicated by the letter "C") and a mail server (indicated by the letter "S") is shown in FIG. 6 for SMTP, FIG. 7 for IMAP and FIG. 8 for POP.

It is necessary for the Identity Provider and mail server to have X.509 certificates for use as TLS server certificates, and for the identity selector, mail application, Identity Provider and mail server to be able to validate these certificates. The Identity Provider and mail server will each generate a public and private key pair, and a security server (34), operating under the control of a network administrator (36), will generate X.509 public key certificates which sign the identity and public key of each of these servers using the private key of the security server.

## **“System for authentication in an electronic mail service” continued**

The public key of the security server will be provided to the database of each client in an X.509 certificate, and be added to the client's trusted certificate authority set.

An illustration of an example implementation of the components of the system for authentication with an Electronic Mail Service is shown in FIG. 12. This system consists of a intranet network switch (400) which is connected to an Internet Service Provider (404) via a firewall router (402). In this implementation, the client components (10) are implemented as operating system and application software deployed on a client workstation computer (406). The mail server (28) for IMAP, POP and SMTP are implemented as server software deployed on a mail server computer (410). The identity provider (20) and its database (22) are implemented as elements of an application server deployed on an identity provider server computer (414). The certification authority (34) is implemented as software deployed on a security server computer (416).

### **Operation - One Embodiment**

The behavior of an electronic mail client when establishing a connection to an electronic mail server is illustrated by the flowchart of FIG. 2A, FIG. 2B, FIG. 2C, FIG. 2D and FIG. 2E. This behavior is applicable to mail clients establishing connections to mail servers using any of the IMAP, POP or SMTP protocols.

An electronic mail client establishes a Transmission Control Protocol (TCP) connection to a mail server (42). The destination port of the TCP connection is determined by the protocol being used for this connection, or by the configuration of the mail system. The mail client and server negotiate the use of Transport Level Security (TLS) or Secure Sockets Layer (SSL) on this

## **“System for authentication in an electronic mail service” continued**

connection (44). The mail client will validate the certificate path sent by the mail server in the TLS or SSL negotiation, and once validated, save the mail server's public key obtained from the mail server's certificate. If this negotiation does not result in the establishment of an association protected against eavesdropping and modification of messages in transit (46), then the mail client will report to the user that there is a server problem (82) and close the connection (88).

After the TLS or SSL association has been established, the electronic mail client will obtain the list of supported SASL mechanisms from the electronic mail server (50). In the SMTP protocol, this is implemented as the mail client sending an EHLO request, and the mail server will respond with an AUTH response with the name of each SASL mechanism the mail server supports. In the IMAP protocol, this is implemented as the electronic mail client sending the CAPABILITY request, and the response by the electronic mail server includes the SASL mechanism names the mail server supports, each preceded by an AUTH= prefix. In the POP protocol, this is implemented as the mail client sending an AUTH request for each of its supported mechanisms in turn, and if the mail server responds with a message prefixed with -ERR, then the mechanism is not supported. If the CARD-INLINE mechanism is not supported by both the mail client and the mail server, but there is another mechanism, such as CRAM-MD5 or DIGEST-MD5, supported by both the mail client and the mail server (70), then that mechanism is used for the authentication (72). Otherwise, if there is no common mechanism, then the electronic mail client will report to the user that there is no common mechanism (74) and close the connection (88).

If both the mail client and the mail server support the CARD-INLINE SASL mechanism (52), then the mail client will initiate the CARD-INLINE negotiation with the mail server (100).

## **“System for authentication in an electronic mail service” continued**

In the SMTP protocol, this is implemented as the mail client sending an AUTH CARD-INLINE request. In the IMAP protocol, this is implemented as the mail client sending an AUTHENTICATE CARD-INLINE request. In the POP protocol, this is implemented as the mail client sending an AUTH CARD-INLINE request. The mail server will specify the mail server's authentication policy requirements in an XML document using the structure specified by WS-SecurityPolicy, encode this document in base64, and send this base64-encoded document in a response to the mail client. The mail client will invoke the identity selector on the client workstation where the mail client is deployed to display to the user information cards which are appropriate to that policy (104). If the user does not have an appropriate card, or does not select a card (106), then the mail client will report an authentication failure to the user (84) and close the connection. Otherwise, the identity selector will establish a connection to the Identity Provider indicated by the card. The identity selector will send to the Identity Provider the public key of the mail server extracted from the mail server's TLS certificate, and authenticate the user to the Identity Provider. If the user is authenticated by the Identity Provider, the Identity Provider will respond to the identity selector with a response that includes a token encrypted for the Relying Party, using the WS-Trust protocol (108). If no token is returned by the Identity Provider to the identity selector (110), then the mail client will report an authentication failure to the user (84) and close the connection to the mail server. Otherwise, the mail client will encode using base64 the token encrypted for the Relying Party returned by the Identity Provider and send this base64-encoded token to the mail server as the next stage of the SASL mechanism CARD-INLINE negotiation (112). The mail server will respond with either a success or failure indication, depending on whether the token was successfully decrypted and the contents

## **“System for authentication in an electronic mail service” continued**

validated. If the mail server does not accept this token (114), then the mail client will report an authentication failure to the user (84) and close the connection to the mail server. Otherwise, the authentication is successful and the mail client will continue with mail transfer (86).

The behavior of an electronic mail server is illustrated by the flowcharts of FIG. 3, and of FIG. 4A, FIG. 4B, FIG. 4C, FIG. 4D and FIG. 4E. This behavior is applicable to mail clients establishing connections for either the IMAP, POP or SMTP protocols.

The connection dispatching thread within a mail server (180) will wait for incoming connections (182). When a new connection is received from a mail client, the connection dispatching thread will create a new connection thread to handle interactions for that connection (184).

A connection thread (202) will send a server banner message to the mail client (204), and then wait for requests from the mail client. If either the time limit of waiting for an incoming request from the mail client is reached, the mail client closes the connection, or the mail client sends a quit command (208), then the connection thread will close the connection (226) and the connection thread will terminate (228). If the mail client sends a request to query the mail server's capabilities (212), the mail server will respond to indicate that the mail server is capable of starting TLS (214). If the mail client sends any request other than a query of capabilities or to start TLS (216), then the mail server will reply with an error (218). Otherwise, the mail client has requested to start TLS, and the mail server will negotiate TLS on the connection to the mail client (220). If the negotiation fails (222), the mail server will close the connection and the connection thread will terminate.

## **“System for authentication in an electronic mail service” continued**

Once TLS has been negotiated on the connection, the mail server then waits for requests from the mail client on that connection, and these requests will be protected from eavesdropping or modification in transit by the underlying TLS protocol (240). If either the time limit of waiting for an incoming request from the mail client is reached (indicating a connection timer has expired), the mail client closes the connection, or the mail client sends a quit command (242), then the connection thread will close the connection (226) and the connection thread will terminate (228). If the mail client sends a request to query the mail server's capabilities (244), the mail server will respond to indicate that the mail server is capable of performing a SASL exchange using the CARD-INLINE mechanism (246). If the mail client sends any request other than a request querying server capabilities or a request to start authentication negotiation (248), then the mail server will reply with an error that authentication is required (250).

In the mail client's request to start authentication negotiation, the mail client will indicate to the mail server its selected SASL mechanism. If the client's requested mechanism is not supported by the mail server (278), then the mail server will reply with an error (280). If the client requests a legacy SASL mechanism and the mail server supports this mechanism for local authentication, then the mail server will use this mechanism to authenticate the user (282). If this authentication failed (286), then the mail server will reply with an error (288), and if there have been repeated authentication failures on this connection (292), then the connection thread will close the connection (226) and the connection thread will terminate (228).

If the mail client requests the CARD-INLINE SASL mechanism (270), then the mail server will send the mail client the server-challenge, consisting of a base64 encoding of an XML-encoded document that specifies mail server's requirements for client authentication (330). The

## **“System for authentication in an electronic mail service” continued**

mail server will then wait for a client-response from the mail client (332). If either the time limit of waiting for an incoming client-response from the mail client is reached (connection timer expired), the mail client closes the connection, or the mail client sends an invalid client-response (334), then the connection thread will close the connection (226) and the connection thread will terminate (228). Otherwise the mail server will parse the received token. The mail server will first undo the base64 encoding to obtain an XML document based on XML Encryption, which contains an encrypted symmetric key, and a token encrypted with that symmetric key. The mail server will next decrypt the symmetric key, using the private key for its TLS certificate's public key. Then the mail server will decrypt the token using this symmetric key (336). The token is a SAML assertion. If the decrypted token is validly encoded, the assertion is from a recognized Identity Provider, and the assertion from the Identity Provider identifies a user authorized to use the mail service, then the user will be treated as authenticated (338), and the mail server will permit the mail client to continue with transfer of email on behalf of the authenticated user. Otherwise, this authentication has failed, and the mail server will reply with an error. If there have been repeated authentication failures on this connection (292), then the connection thread will close the connection (226) and the connection thread will terminate (228).

### **DESCRIPTION - Another Embodiment**

As another embodiment, this disclosure defines a mechanism by which the mail server may leverage a mail system Security Token Service (STS) server to validate the token returned by the Identity Provider and transform this token into a token format accepted by the mail server. This alternative embodiment is useful in environments in which the mail servers are only capable of recognizing a single SAML assertion token format and issuer, and the validation of the

## **“System for authentication in an electronic mail service” continued**

Identity Provider identity and token formats is offloaded to a separate server from the mail server.

The mail system STS server is a server for web services, implementing the WS-Trust and WS-MetadataExchange protocols with a transport of HTTPS (HTTP over TLS).

In order to support this alternative embodiment, this disclosure defines a new SASL mechanism, CARD-RPSTS. The name of this CARD-RPSTS SASL mechanism is "CARD-RPSTS". The mechanism is server first, does not transfer an authorization identity string, does not offer a security layer, and requires the use of an underlying integrity layer, Transport Layer Security (TLS).

In the CARD-RPSTS SASL mechanism, the server-challenge is sent from the mail server to the mail client. The server-challenge consists of the Uniform Resource Identifier (URI) of the mail system STS responder for WS-MetadataExchange. This URI must be of the "https" protocol.

The client-response in the CARD-RPSTS mechanism is a protocol data unit sent from the mail client to the mail server. The client-response consists of a base64 encoding of an XML-encoded document, representing an encrypted authentication token from the mail system STS. The token is a Security Assertion Markup Language (SAML) assertion, in a format defined in the document "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", edited by Scott Cantor, John Kemp, Rob Philpott and Eve Maler. The token is encrypted by a randomly generated symmetric key, and this key is encrypted with the public key of the mail server, obtained from the TLS server certificate sent to the mail client by the mail

## **“System for authentication in an electronic mail service” continued**

server. The encryption is performed according to the specification contained in the document "XML Encryption Syntax and Processing" by Donald Eastlake, Joseph Reagle, Takeshi Imamura, Blair Dillaway, and Ed Simon.

An illustration of this SASL exchange between a mail client (indicated by the letter "C") and a mail server (indicated by the letter "S") is shown in FIG. 9 for SMTP, FIG. 10 for IMAP and FIG. 11 for POP.

It is necessary for the mail system STS server to have an X.509 certificate for use as a TLS server certificate, and for the identity selector, mail application and mail server to be able to validate this certificate. The mail system STS server will generate a public and private key pair, and a security server (34), operating under the control of a network administrator (36), will generate a X.509 public key certificate which sign the identity and public key of this server using the private key of the security server.

An illustration of an example implementation of the components of the system for authentication with an Electronic Mail Service is shown in FIG. 12. In this implementation, the mail system STS server (26) is implemented as a collection of servlets running within an application server container deployed on a mail system STS server computer (412).

### **Operation - Another Embodiment**

The behavior of an electronic mail client when establishing a connection to an electronic mail server is illustrated by the flowchart of FIG. 2A, FIG. 2B, FIG. 2C, FIG. 2D and FIG. 2E. This behavior is applicable to mail clients establishing connections to mail servers using any of the IMAP, POP or SMTP protocols.

## **“System for authentication in an electronic mail service” continued**

An electronic mail client establishes a Transmission Control Protocol (TCP) connection to a mail server (42). The destination port of the TCP connection is determined by the protocol being used for this connection, or by the configuration of the mail system. The mail client and server negotiate the use of Transport Level Security (TLS) or Secure Sockets Layer (SSL) on this connection (44). The mail client will validate the certificate path sent by the mail server in the TLS or SSL negotiation, and once validated, save the mail server's public key obtained from the mail server's certificate. If this negotiation does not result in the establishment of an association protected against eavesdropping and modification of messages in transit (46), then the mail client will report to the user that there is a server problem (82) and close the connection (88).

After the TLS or SSL association has been established, the electronic mail client will obtain the list of supported SASL mechanisms from the electronic mail server (50). In the SMTP protocol, this is implemented as the mail client sending an EHLO request, and the mail server will respond with an AUTH response with the name of each SASL mechanism the mail server supports. In the IMAP protocol, this is implemented as the electronic mail client sending the CAPABILITY request, and the response by the electronic mail server includes the SASL mechanism names the mail server supports, each preceded by an AUTH= prefix. In the POP protocol, this is implemented as the mail client sending an AUTH request for each of its supported mechanisms in turn, and if the mail server responds with a message prefixed with -ERR, then the mechanism is not supported. If the CARD-RPSTS mechanism is not supported by both the mail client and the mail server, but there is another mechanism, such as CRAM-MD5 or DIGEST-MD5, supported by both the mail client and the mail server (70), then that mechanism is used for the authentication (72). Otherwise, if there is no common mechanism,

## **“System for authentication in an electronic mail service” continued**

then the electronic mail client will report to the user that there is no common mechanism (74) and close the connection (88).

If both the mail client and the mail server support the CARD-RPSTS SASL mechanism (56), then the mail client will initiate the CARD-RPSTS negotiation with the mail server (130). In the SMTP protocol, this is implemented as the mail client sending an AUTH CARD-RPSTS request. In the IMAP protocol, this is implemented as the mail client sending an AUTHENTICATE CARD-RPSTS request. In the POP protocol, this is implemented as the mail client sending an AUTH CARD-RPSTS request. The mail server will send the URI of the mail system STS server in response to the mail client as the server-challenge (132). The mail client will establish a second connection to the mail system STS server using HTTP over TLS (134). If the mail system STS server is unavailable (136), then the mail client will report to the user that there is a server problem (82) and close the connection (88). Otherwise, the mail client will obtain the mail system policy from the mail system STS server using WS-MetadataExchange (138).

The mail client will invoke the identity selector on the workstation where the mail client is deployed to display to the user information cards which are appropriate to the mail system policy returned by the mail system STS server (140). If the user does not have an appropriate card or does not select a card (160), then the mail client will report an authentication failure to the user (84) and close the connections. Otherwise, the identity selector will establish a third connection, to the Identity Provider indicated by the card. The identity selector will authenticate the user to the Identity Provider, and send the public key of the mail system STS server. If the Identity Provider authenticates the user, the Identity Provider will reply to the identity selector

## **“System for authentication in an electronic mail service” continued**

using the WS-Trust protocol with a token encrypted for the mail system STS server (162). If no token is available (164), then the mail client will report an authentication failure to the user (84) and close the connections to the mail server and mail system STS server. Otherwise, the mail client will send this token encrypted for the mail system STS server to the mail system STS server using the WS-Trust protocol (166) and await a response token (168). If the mail system STS server rejects this token, and no response token is available (170), then the mail client will report an authentication failure to the user (84) and close the connections to the mail server and mail system STS server. Otherwise, the mail client will base64 encode the response token returned by the mail system STS server and send the base64 encoded response token to the mail server (112). The mail server will respond with either a success or failure indication, depending on whether the response token from the mail system STS server was successfully decrypted and the contents validated. If the mail server does not accept this token (114), then the mail client will report an authentication failure to the user (84) and close the connection to the mail server. Otherwise, the authentication is successful and the mail client will continue with mail transfer (86).

The behavior of an electronic mail server is illustrated by the flowcharts of FIG. 3, and of FIG. 4A, FIG. 4B, FIG. 4C, FIG. 4D and FIG. 4E. This behavior is applicable to mail clients establishing connections for either the IMAP, POP or SMTP protocols.

The connection dispatching thread within a mail server (180) will wait for incoming connections (182). When a new connection is received from a mail client, the connection dispatching thread will create a new connection thread to handle interactions for that connection (184).

## **“System for authentication in an electronic mail service” continued**

A connection thread (202) will send a server banner message to the mail client (204), and then wait for requests from the mail client. If either the time limit of waiting for an incoming request from the mail client is reached, the mail client closes the connection, or the mail client sends a quit command (208), then the connection thread will close the connection (226) and the connection thread will terminate (228). If the mail client sends a request to query the mail server's capabilities (212), the mail server will respond to indicate that the mail server is capable of starting TLS (214). If the mail client sends any request other than a query of capabilities or to start TLS (216), then the mail server will reply with an error (218). Otherwise, the mail client has requested to start TLS, and the mail server will negotiate TLS on the connection to the mail client (220). If the negotiation fails (222), the mail server will close the connection and the connection thread will terminate.

Once TLS has been negotiated on the connection, the mail server then waits for requests from the mail client on that connection, and these requests will be protected from eavesdropping or modification in transit by the underlying TLS protocol (240). If either the time limit of waiting for an incoming request from the mail client is reached (indicating a connection timer has expired), the mail client closes the connection, or the mail client sends a quit command (242), then the connection thread will close the connection (226) and the connection thread will terminate (228). If the mail client sends a request to query the mail server's capabilities (244), the mail server will respond to indicate that the mail server is capable of performing a SASL exchange using the CARD-RPSTS mechanism (246). If the mail client sends any request other than a request querying server capabilities or a request to start authentication negotiation (248), then the mail server will reply with an error that authentication is required (250).

## **“System for authentication in an electronic mail service” continued**

In the mail client's request to start authentication negotiation, the mail client will indicate to the mail server its selected SASL mechanism. If the client's requested mechanism is not supported by the mail server (278), then the mail server will reply with an error (280). If the client requests a legacy SASL mechanism and the mail server supports this mechanism for local authentication, then the mail server will use this mechanism to authenticate the user (282). If this authentication failed (286), then the mail server will reply with an error (288), and if there have been repeated authentication failures on this connection (292), then the connection thread will close the connection (226) and the connection thread will terminate (228).

If the mail client requests the CARD-RPSTS SASL mechanism (270), then the mail server will send the mail client a server-challenge, the HTTPS URI of the mail system STS server (340). The mail server will then wait for a client-response from the mail client (342). If either the time limit of waiting for an incoming client-response from the mail client is reached (connection timer expired), the mail client closes the connection, or the mail client sends an invalid client-response (344), then the connection thread will close the connection (226) and the connection thread will terminate (228). Otherwise the mail server will parse the received token. The mail server will first undo the base64 encoding to obtain an XML document based on XML Encryption, which contains an encrypted symmetric key, and a token encrypted with that symmetric key. The mail server will next decrypt the symmetric key, using the private key for its TLS certificate's public key. Then the mail server will decrypt the token using this symmetric key (346). The token is a SAML assertion. If the decrypted token is validly encoded, the assertion is from the mail system STS server, and the assertion from the mail system STS server identifies a user authorized to use the mail service, then the user will be treated as authenticated

## **“System for authentication in an electronic mail service” continued**

(348), and the mail server will permit the mail client to continue with transfer of email on behalf of the authenticated user. Otherwise, this authentication has failed, and the mail server will reply with an error. If there have been repeated authentication failures on this connection (292), then the connection thread will close the connection (226) and the connection thread will terminate (228).

The behavior of a mail system STS server is illustrated by the flowchart of FIG. 5. The mail system STS server will wait for a HTTPS connection and a SOAP-formatted request from the electronic mail client (362). The mail system STS server will parse the request to determine the identity of the client sending the request, the type of the request and its parameters (364). If the request is a WS-MetadataExchange request for the security policy (366), then the mail system STS server will reply to the mail client with the mail system STS server's security policy (368). Otherwise, the request is a WS-Trust request, which contains the public key of the mail server.

The WS-Trust request encloses an XML document based on XML Encryption, which contains an encrypted symmetric key, and a token encrypted with that symmetric key. The mail system STS server will decrypt the symmetric key, using the private key for its TLS certificate's public key. Then the mail system STS server will decrypt the token using this symmetric key (370). The token is a SAML assertion. If the decrypted token is not validly encoded, the assertion is not from a recognized Identity Provider, or the assertion from the Identity Provider does not identify a user authorized to use the mail service (372), then the mail system STS server will reply with an error (374). Otherwise, the mail system STS server will build a response token (376). The response token is a SAML assertion. The mail system STS server generates a new random symmetric key, encrypts the response token with the symmetric key, and encrypts the

## **“System for authentication in an electronic mail service” continued**

symmetric key with the public key of the mail server. The mail system STS server replies to the mail client with the encrypted key and encrypted token (378), which the mail client will send to the mail server in the SASL CARD-RPSTS client-response.

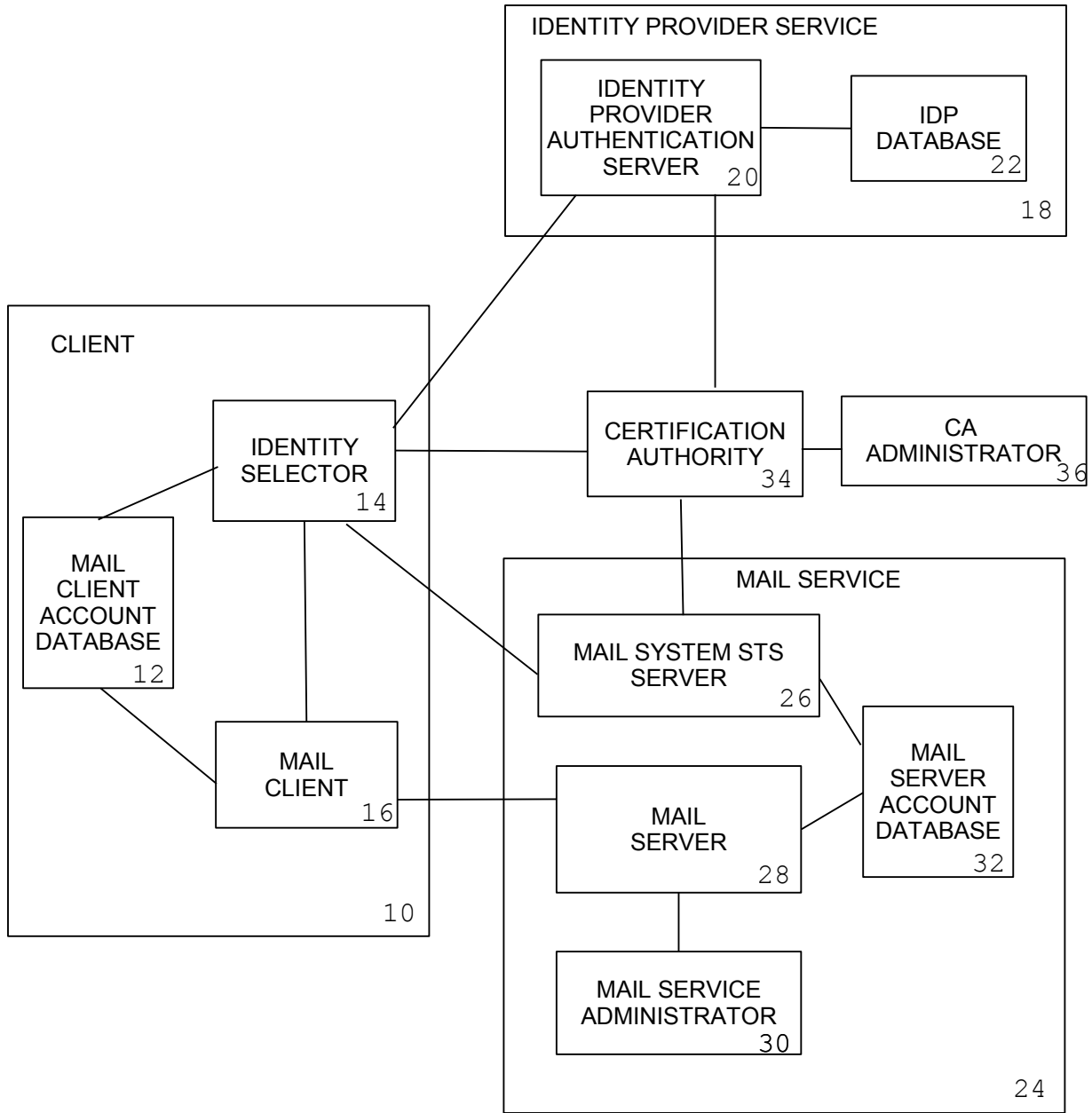


FIG. 1

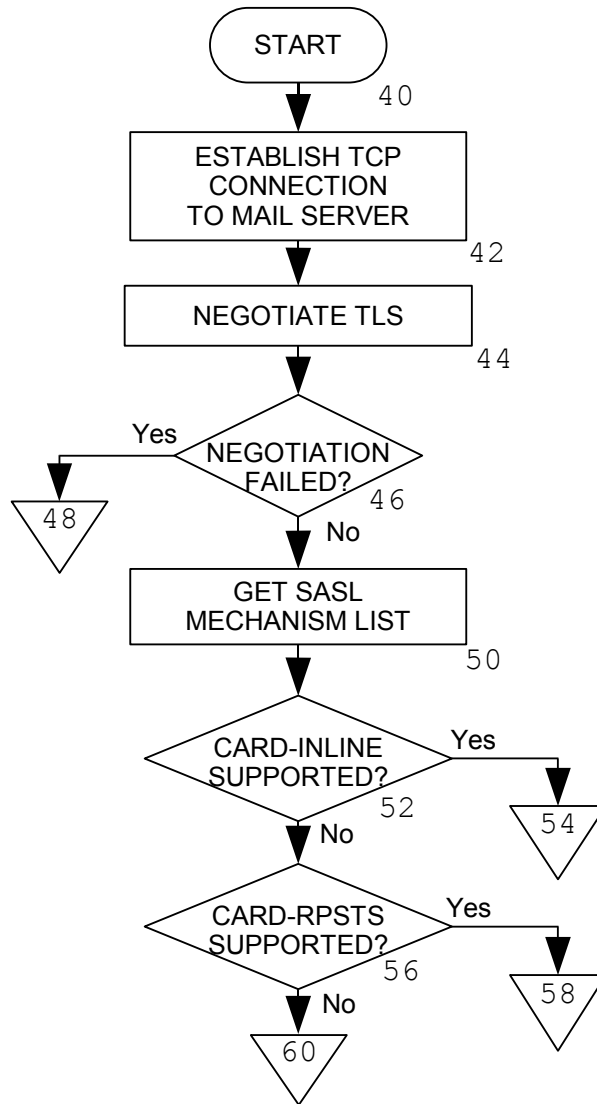


FIG. 2A

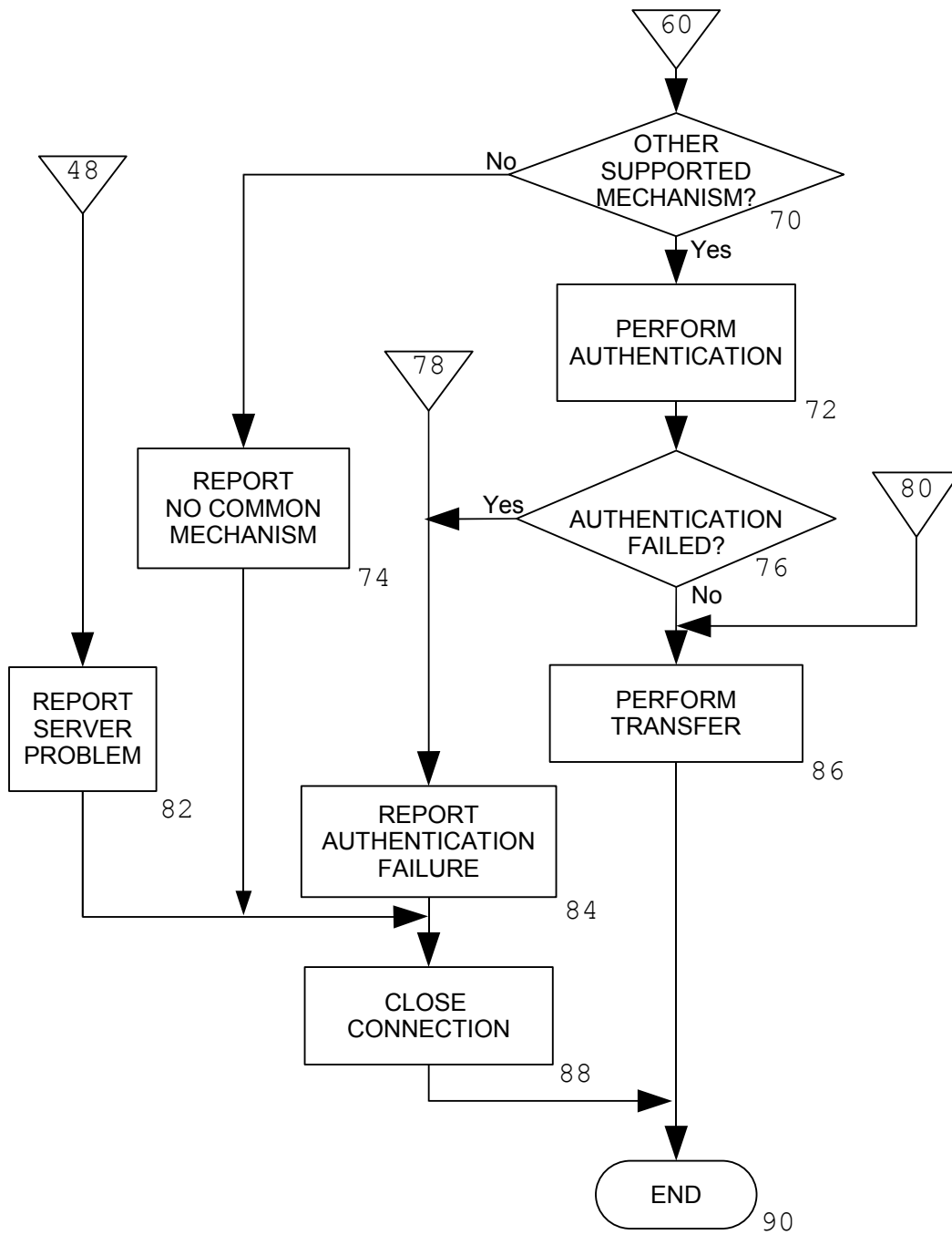


FIG. 2B

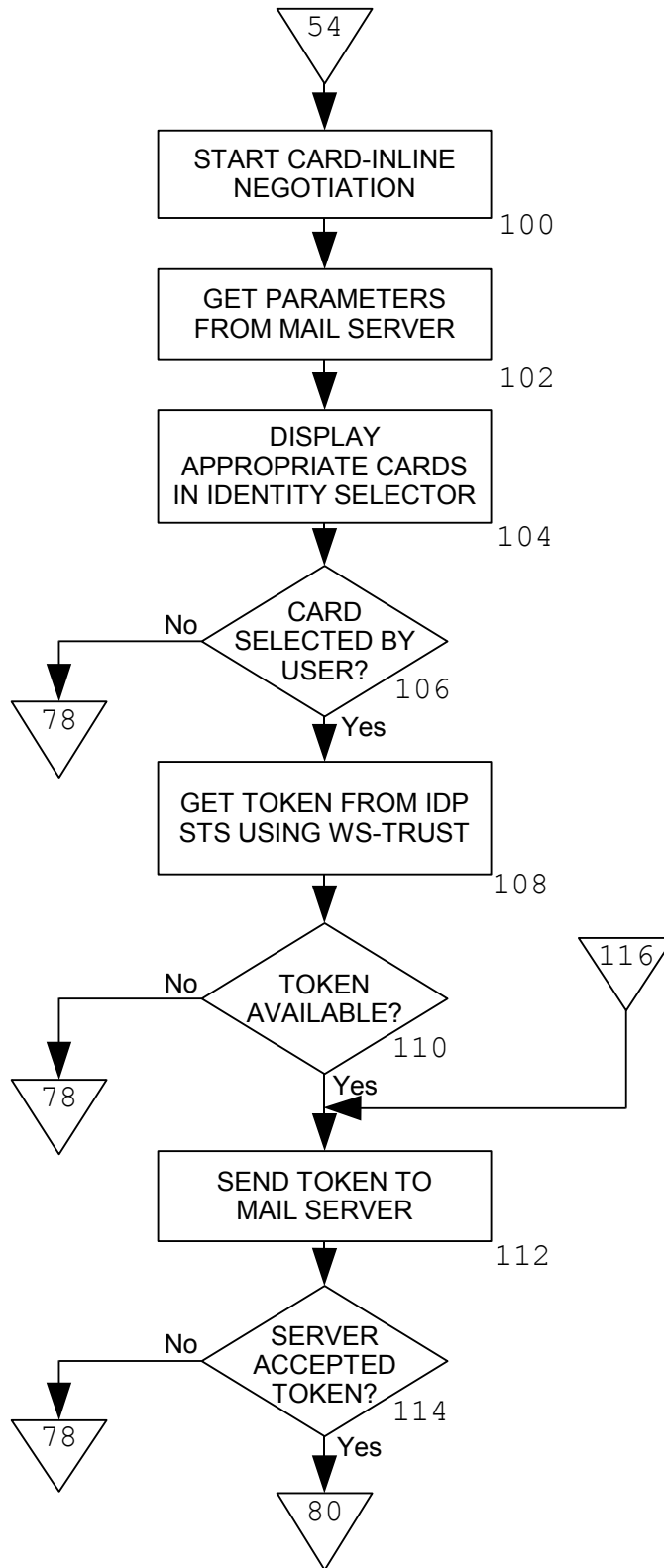


FIG. 2C

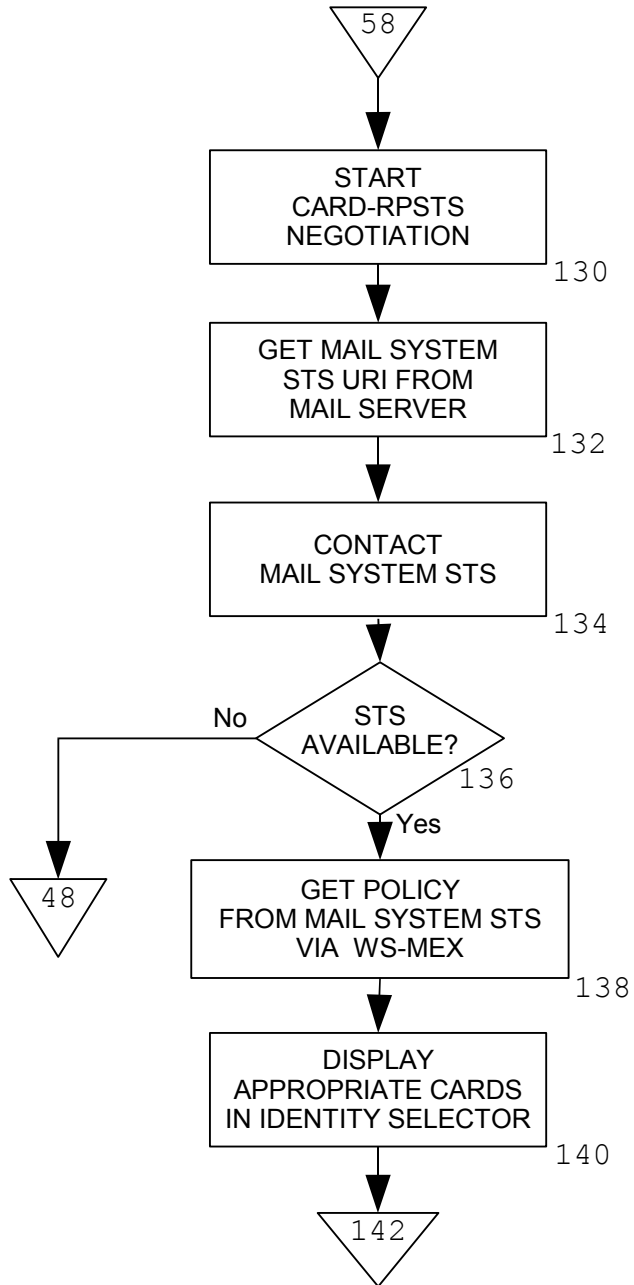


FIG. 2D

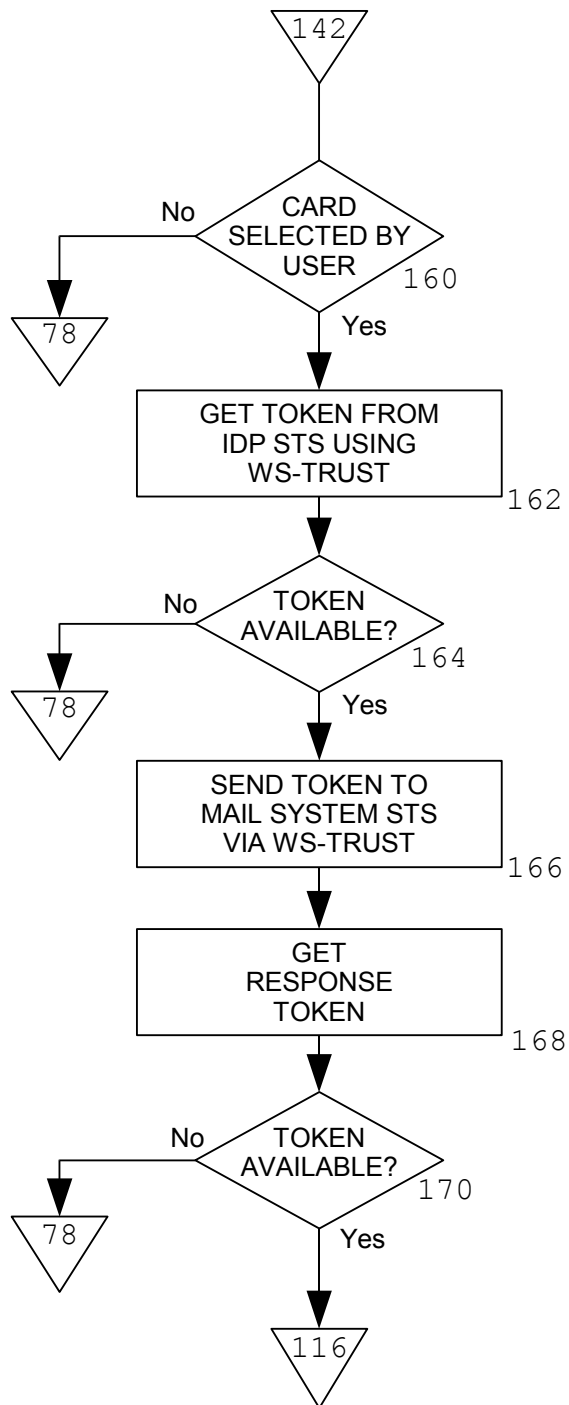
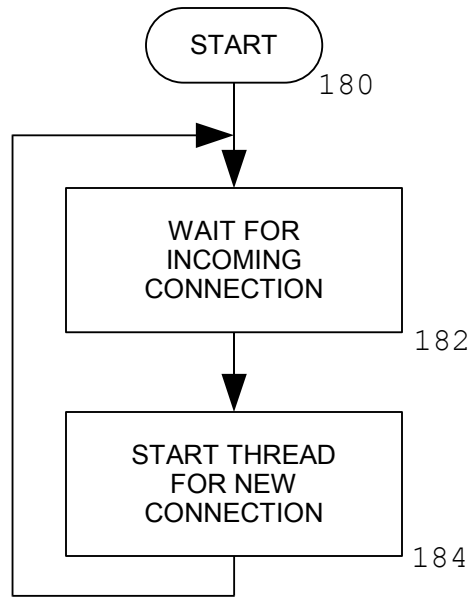


FIG. 2E



**FIG. 3**

Copyright 2008 Informed Control Inc.

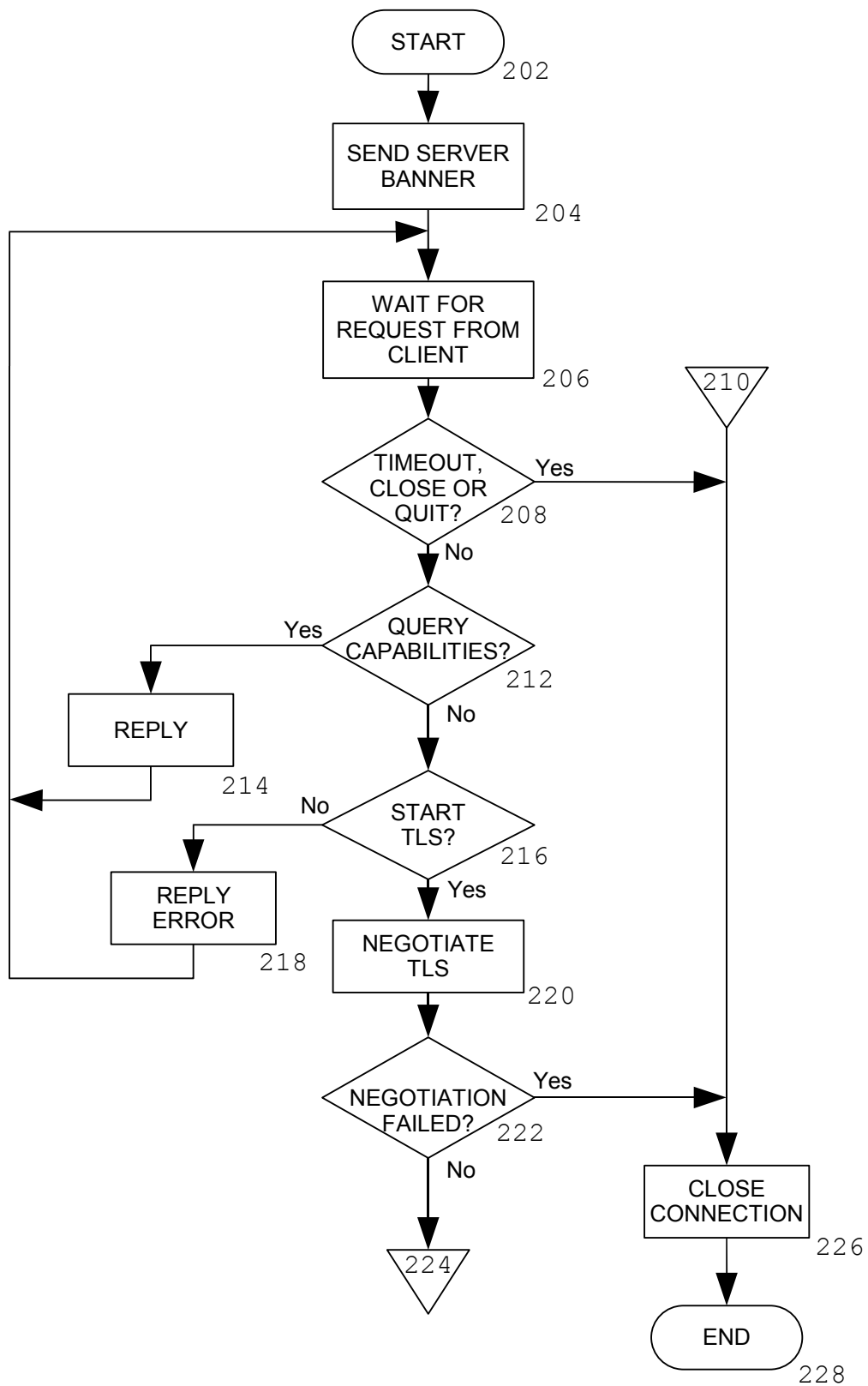


FIG. 4A

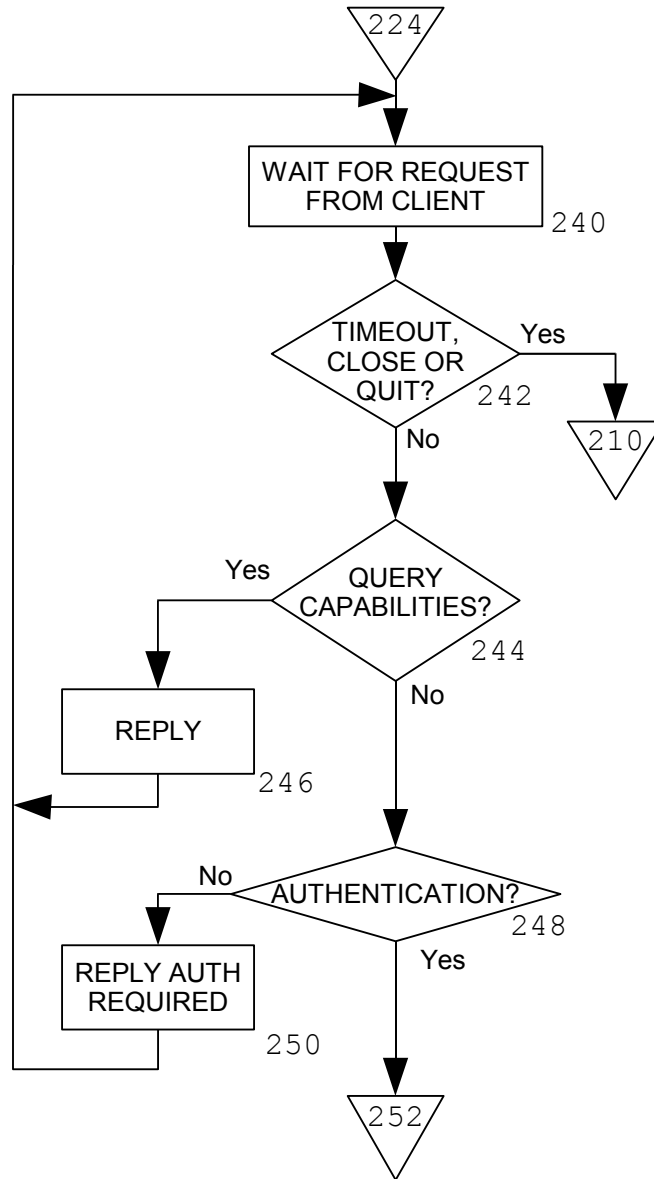


FIG. 4B

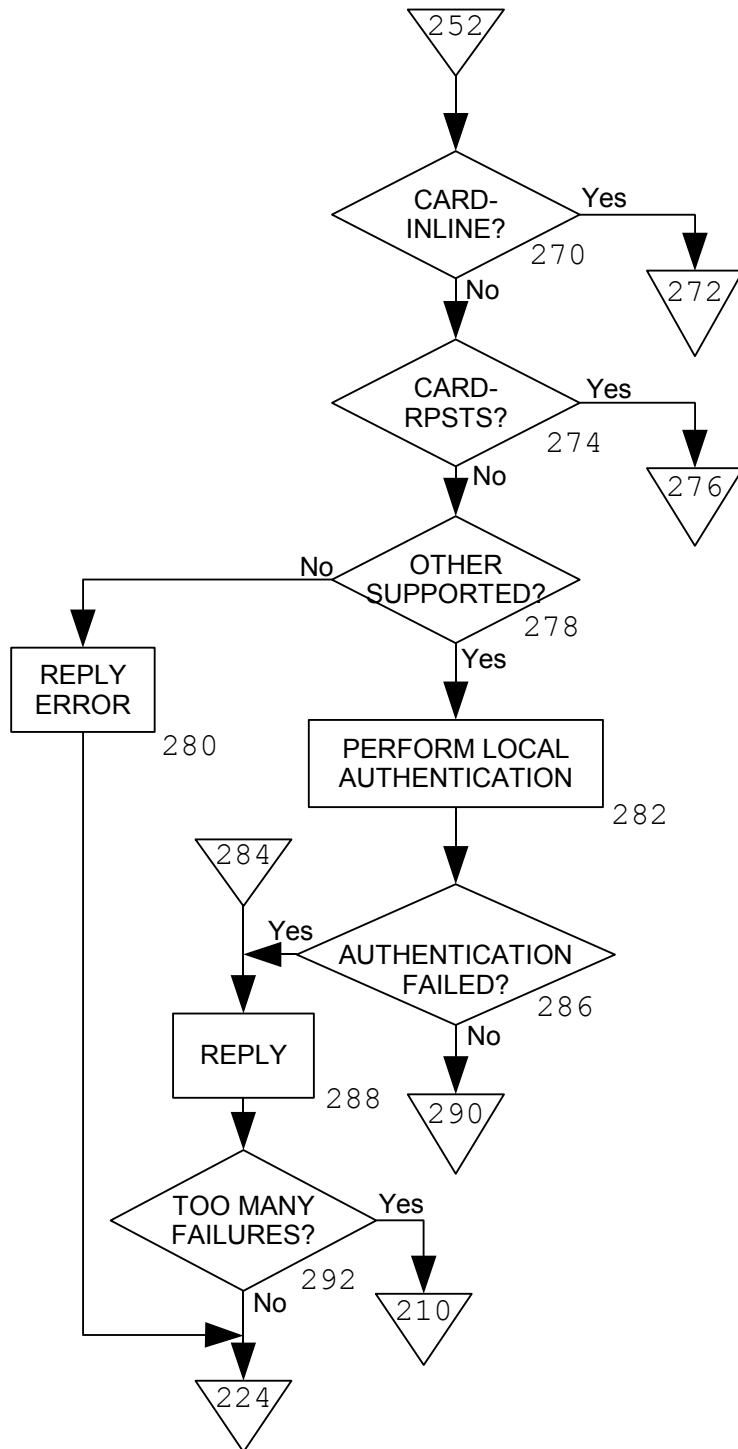


FIG. 4C

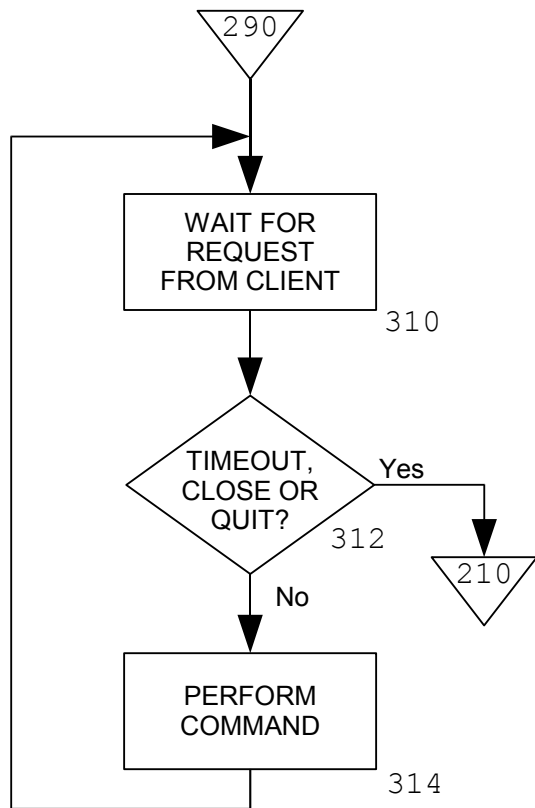


FIG. 4D

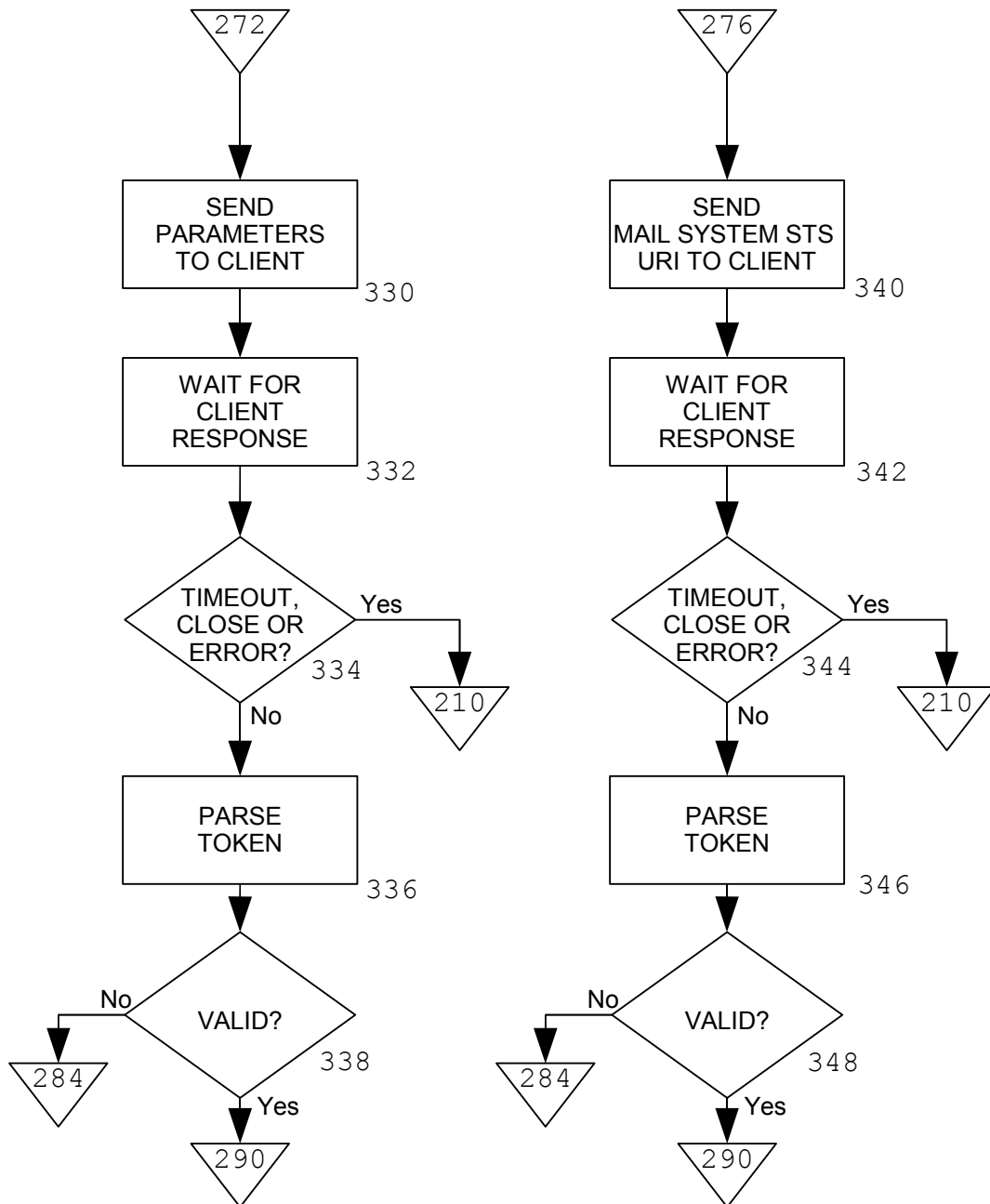
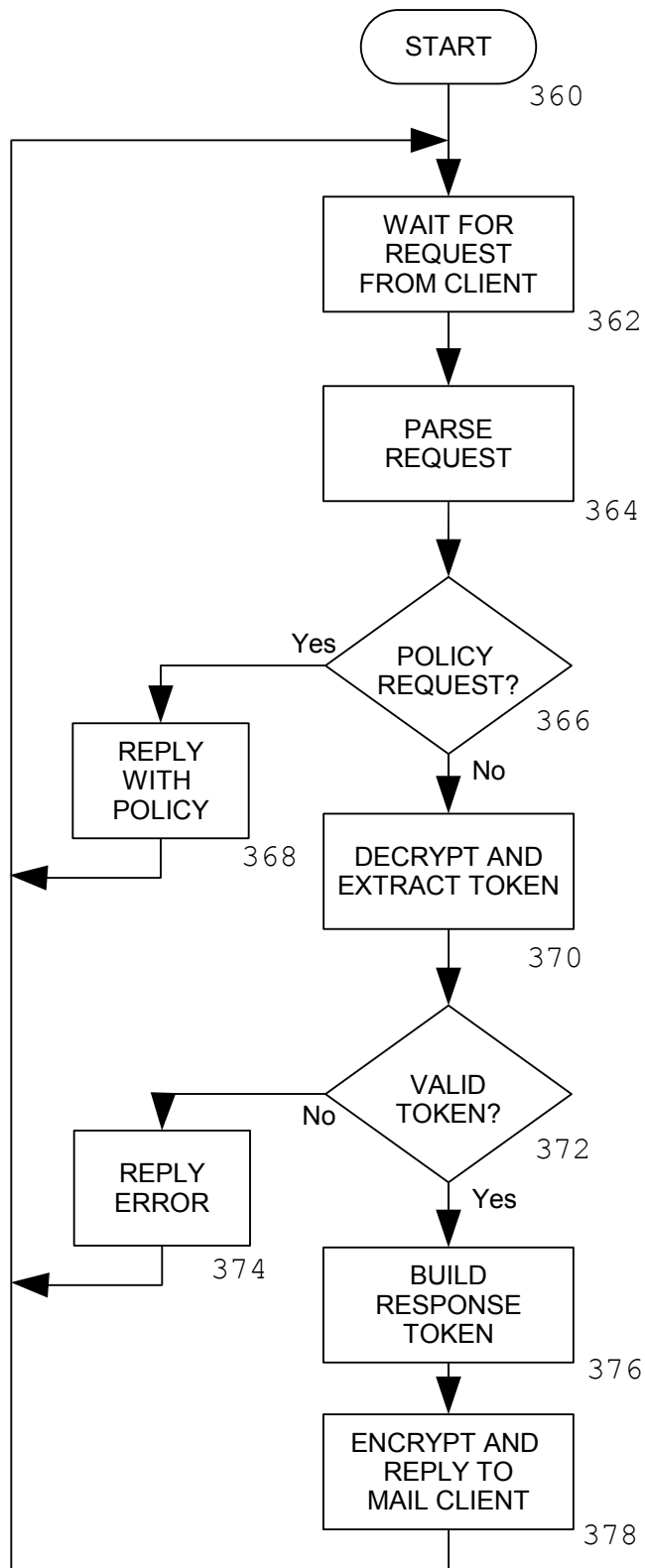


FIG. 4E



**FIG. 5**

## SMTP

### *CLIENT OPENS CONNECTION*

S: 220 SERVER.EXAMPLE.COM SMTP SERVICE READY

C: EHLO CLIENT.EXAMPLE.ORG

S: 250-SERVER.EXAMPLE.COM READY

S: 250 STARTTLS

C: STARTTLS

S: 220 GO AHEAD

### *TLS NEGOTIATION*

C: EHLO CLIENT.EXAMPLE.ORG

S: 250-SERVER.EXAMPLE.COM READY

S: 250 AUTH CARD-INLINE

C: AUTH CARD-INLINE

S: 334 *BASE64-ENCODED POLICY*

C: *BASE64-ENCODED TOKEN*

S: 235 AUTHENTICATION SUCCESSFUL

FIG. 6

## IMAP

### *CLIENT OPENS CONNECTION*

S: \* OK IMAP4 SERVER

C: A001 STARTTLS

S: A001 OK BEGIN TLS NEGOTIATION

### *TLS NEGOTIATION*

C: A002 CAPABILITY

S: \* CAPABILITY IMAP4REV1 AUTH=CARD-INLINE

S: A002 OK CAPABILITY COMPLETED

C: A003 AUTHENTICATE CARD-INLINE

S: + **BASE64-ENCODED POLICY**

C: **BASE64-ENCODED TOKEN**

S: A003 OK AUTHENTICATION SUCCESSFUL

FIG. 7

## POP

### *CLIENT OPENS CONNECTION*

S: +OK POP3 SERVER READY

C: STLS

S: +OK BEGIN TLS NEGOTIATION

### *TLS NEGOTIATION*

C: AUTH CARD-INLINE

S: + **BASE64-ENCODED POLICY**

C: **BASE64-ENCODED TOKEN**

S: +OK AUTHENTICATION SUCCESSFUL

FIG. 8

## SMTP

### *CLIENT OPENS CONNECTION*

S: 220 SERVER.EXAMPLE.COM SMTP SERVICE READY

C: EHLO CLIENT.EXAMPLE.ORG

S: 250-SERVER.EXAMPLE.COM READY

S: 250 STARTTLS

C: STARTTLS

S: 220 GO AHEAD

### *TLS NEGOTIATION*

C: EHLO CLIENT.EXAMPLE.ORG

S: 250-SERVER.EXAMPLE.COM READY

S: 250 AUTH CARD-RPSTS

C: AUTH CARD-RPSTS

S: 334 *MAIL SYSTEM STS URI*

C: *BASE64-ENCODED TOKEN FROM MAIL SYSTEM STS*

S: 235 AUTHENTICATION SUCCESSFUL

## FIG. 9

## IMAP

### *CLIENT OPENS CONNECTION*

S: \* OK IMAP4 SERVER

C: A001 STARTTLS

S: A001 OK BEGIN TLS NEGOTIATION

### *TLS NEGOTIATION*

C: A002 CAPABILITY

S: \* CAPABILITY IMAP4REV1 AUTH=CARD-RPSTS

S: A002 OK CAPABILITY COMPLETED

C: A003 AUTHENTICATE CARD-RPSTS

S: + *MAIL SYSTEM STS URI*

C: *BASE64-ENCODED RESPONSE TOKEN FROM MAIL SYSTEM STS*

S: A003 OK AUTHENTICATION SUCCESSFUL

FIG. 10

## POP

### *CLIENT OPENS CONNECTION*

S: +OK POP3 SERVER READY

C: STLS

S: +OK BEGIN TLS NEGOTIATION

### *TLS NEGOTIATION*

C: AUTH CARD-RPSTS

S: + *MAIL SYSTEM STS URI*

C: *BASE64-ENCODED RESPONSE TOKEN FROM MAIL SYSTEM STS*

S: +OK AUTHENTICATION SUCCESSFUL

FIG. 11

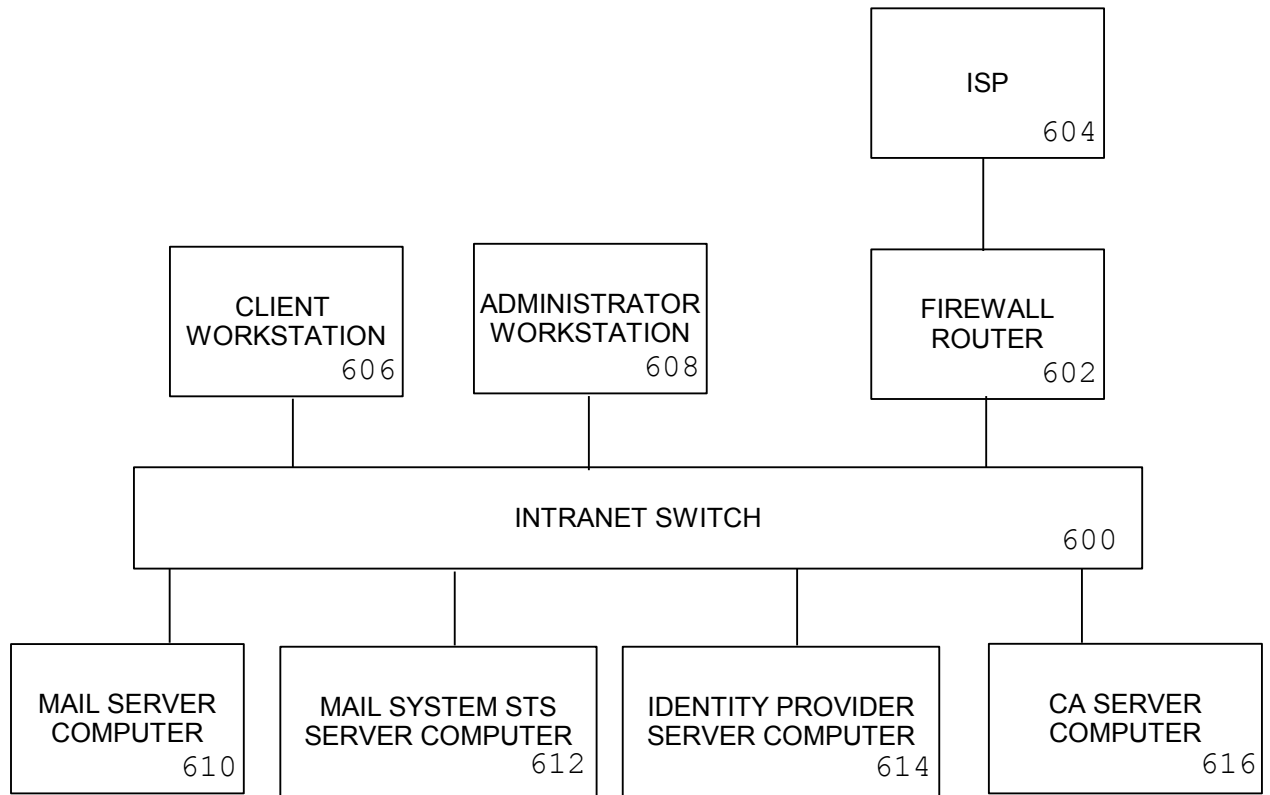


FIG. 12