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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 December 2006 (07.12.2006)

PCT

(10) International Publication Number
WO 2006/128503 A1

(51) International Patent Classification:
H04L 29/06 (2006.01)

(21) International Application Number:
PCT/EP2006/001681

(22) International Filing Date:
21 February 2006 (21.02.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
S2005/0376 3 June 2005 (03.06.2005) IE

(71) Applicant *(for all designated States except US)*: **ASAVIE R & D LIMITED** [IE/IE]; 24 Herbert Lane, Dublin 2 (IE).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **MAHER, Thomas** [IE/IE]; 14 Westminster Lawns, Dublin 18 (IE).

(74) Agents: **BOYCE, Conor** et al.; F. R. Kelly & Co., 27 Clyde Road, Ballsbridge, Dublin 4 (IE).

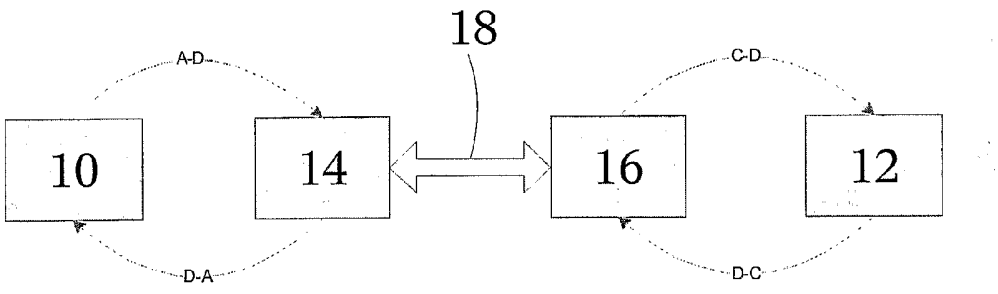
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

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

(54) **Title:** SECURE NETWORK COMMUNICATION SYSTEM AND METHOD



(57) **Abstract:** A secure network communication system and method for secure data exchange using transmission control protocol are disclosed. The system provides for data exchange using between a client and a server, by way of an agent and a broker interconnected to exchange data over an unsecured network link. Upon receipt of a control packet from the client, the broker forwards a modified control packet to the agent using a secure protocol. The agent then inspects the modified control packet and forwards it to the server. Upon receipt of a response packet from the server, the agent forwards the response packet to the broker using a secure protocol and upon receipt of the response packet, the agent modifies the response packet and forwards it to the client. In the case that the exchange of control packets indicates establishment of a TCP session, the agent and the broker establish a data channel between themselves to create a transparent TCP channel between the client and the server.

WO 2006/128503 A1

Secure network communication system and method

This invention relates to a secure network communication system and method. In particular, it relates a communication system and method to enable secure clients to obtain secure and transparent access to a remote server over an insecure network.

It is widely recognised that connecting a computer to communicate with an insecure network such as the Internet represents a security risk. In order to perform useful tasks, such a computer must necessarily react to data that is received from remote servers over the network. Malicious attackers can exploit deficiencies in the computer's operating system and applications to react in such a way as to perform functions that might compromise the security or serviceability of the computer. Accordingly, it is normal for a private computer or network of computers is not connected directly to a public network. Rather, a connection is made to the network through some restrictive interface that controls the nature of the data that can be exchanged between the private computers and the unsecured network. The typical configuration of the hosts and networks is that client hosts are "inside" or trusted, while servers are "outside" or untrusted.

Two existing methods for enabling secure transparent gateway access between networks are by means of a proxy (for example, as disclosed in US-A-5 623 601 and US-A-5 781 550) and by network address translation (NAT) (for example, as disclosed in US-A-5 793 763). In many cases, these arrangements provide an adequate service to "inside" users and computers. However, the service provided it is not the same as a direct network connection. In the case of a proxy, normally only connection to a configurable range of well-known ports on remote server are allowed. This can allow a network administrator to restrict the types of service a user of the network can access, and the client machines are hidden from the outside servers.

These known systems allow a network user to access outside services such as the Worldwide Web, Internet e-mail, and so forth, and so are selectively transparent to OSI application layer (Layer 7) protocols. However, they do not provide transparent connections to lower-level network protocols. In particular, at the OSI transport layer (Layer 4), at which the Transport Control Protocol (TCP) is commonly used and at the OSI network layer (Layer 3), at which the Internet Protocol (IP) is commonly used, these systems lack transparency. This can be a source of problems when the network is not operating normally, as, for example, when a TCP establishment fails.

The known implementation of a transparent proxy gateway is not fully transparent, particularly in regard to the TCP/IP control packets. For example, consider the case that a client completes the establishment of the connection between itself and a proxy before the proxy attempts to establish a connection to the destination host. If the proxy connection fails, then the established connection to the host is disconnected. The client sees therefore an incorrect view of the network, which for service discovery by clients of server services is of material importance. In this example, the client perceives the service port as open, which is not the case.

More recent developments in remote access have produced what are sometimes described as reverse proxies or gateways, where clients "outside" the network are authenticated and provided secure access to "inside" servers. However, the gateways are not transparent, requiring instead client-deployed software, which negotiates connections with the reverse proxy.

An aim of this invention is to provide increased transparency, as compared with known systems, while maintaining security for the client systems. It is a particular aim of preferred embodiments of the invention to provide full TCP transparency at the client to permit automatic discovery of network resources.

From a first aspect, this invention provides a network gateway through which a client can communicate with a remote host using transport control protocol (TCP), in which data transmission system for secure data exchange using transmission control protocol between a client and a server, comprising an agent and a broker connected to exchange data over an unsecured network link, in which: upon receipt of a control packet from the client, the broker forwards a modified control packet to the agent using a secure

protocol; the broker inspects the modified control packet and forwards it to the server; upon receipt of a response packet from the server, the agent forwards the response packet to the broker using a secure protocol; and upon receipt of the response packet, the broker modifies the response packet and forwards it to the client; wherein in the case that the exchange of control packets indicates establishment of a TCP session, the broker and the agent establish a data channel between themselves to create a transparent TCP channel between the client and the server.

Typically, the server is connected to the broker for exchange of data over a secured network – that is, an "inside" network. To protect the inside network, the secured network is typically connected to the unsecured network by one or more of a firewall, a proxy or a network address translation (NAT) device.

The secure protocol may employ a transport layer security (TLS) protocol. The TLS protocol may be used to carry secure hypertext transport protocol (HTTPS). This has the advantage that port 443, the port used for HTTPS, is typically left open on network firewalls and proxies. Alternatively, the HTTP CONNECT protocol may be used.

The control packets handled by the system include TCP SYN packets and TCP ACK packets. The response packets include TCP SYN ACK and TCP RST ACK packets. These are the packets principally involved in setting up a TCP session. The control packet and the response packet may be Internet Control Message Protocol packets. Such packets are typically involved in automatic discovery of network resources.

By suitable modification of the source and destination addresses within these packets, the data exchanged by the client and the server is similar to that which would be exchanged if the client and server were in direct communication.

In order to control the services and servers that a user can access, the broker applies rules to determine whether or not the control packet is forwarded to the agent. The rules may depend upon one or more of the originator of the packet, the target address of the packet, and the target port number of the packet.

Embodiments of the invention may be operative to service multiple virtual networks through a client access network. In such embodiments, the client access network is typically responsible for managing client access. The client access network may use

one or both of standard AAA (Authorisation, Authentication and Accounting) and VPN services.

From a second aspect, the invention provides a data transmission system for secure data exchange using transmission control protocol between a client and a server, comprising an agent and a broker connected to exchange data over an unsecured network link, in which: upon receipt of a control packet from the client, the broker forwards a modified control packet to the agent using a secure protocol; the broker inspects the modified control packet and forwards it to the server; upon receipt of a response packet from the server, the agent forwards the response packet to the broker using a secure protocol; and upon receipt of the response packet, the broker modifies the response packet and forwards it to the client; wherein in the case that the exchange of control packets indicates establishment of a TCP session, the broker and the agent establish a data channel between themselves to create a transparent TCP channel between the client and the server.

An embodiment of the invention will now be described in detail, by way of example, with reference to the accompanying drawings, in which:

Figure 1 is a diagram that illustrates communication between a client and a server in a system embodying the invention;

Figure 2 is a block diagram of a broker being a component of a system embodying the invention; and

Figure 3 is a block diagram of an agent being a component of a system embodying the invention.

With reference first to Figure 1, a client 10 operating inside a secure network connects to an outside remote server 12 by way of a broker 14 and an agent 16. The broker 14 and agent 16 communicate over an insecure network 18, most usually including an Internet link.

In the system of Figure 1, the client 10 is attempting to establish communication with the server 12. The broker 14 serves as an intervening Internet gateway, which "spoofs" the server 12. The broker 14 co-ordinates with the remote agent 16, using a secure

application layer protocol. The agent 16 connects to the server 12 and acts as a client to the broker, and so must follow the broker's protocols, the nature of which will become apparent from the following description. Data from the client 10 to the server 12 therefore flows in connections from the client 10 to the broker 14, where they are routed and securely transported to the agent 16, where the data is carried in onward connections from the agent 16 to the server 12. The agents 16 connect to the broker 14 using a secure transport, such as TLS, in a manner that facilitates the traversal of intervening firewalls, proxies and NAT devices. The preferred approach is to use port 443, typically allowed for HTTPS traffic to secure Internet web servers.

The method used by the broker 14 and agent 16 to establish a transparent connection at the transport layer using TCP between the client 10 and the server 12 will now be described. In this embodiment (as is most typical) the transport layer protocol is the Transport Control Protocol (TCP).

As is well-known to those skilled in networking technology, a TCP session between a client 10 and a server 12 is set up by a client sending a synchronisation SYN packet to the server 12 and waiting for an acknowledgement ACK packet in return from the server 12. When the client 10 receives the ACK packet, it sends an ACK packet in response to the server 12. The method by which the broker 14 of this embodiment operates works by capturing TCP SYN packets at the IP layer and holding them while routing and policy decisions are made. If an agent-route for the TCP connection is found, the connection is offered to that agent 16, which can then attempt to make the connection, by generating its own TCP SYN packet. The agent 16 will get in response from the server one of: TCP SYN ACK, meaning that the connection is established; TCP RST ACK, meaning that the connection is refused; or ICMP packet, meaning that the network, host or port is unreachable.

If the connection succeeds, the agent 16 establishes a data session for the connection traffic to be transferred between broker and agent, and the agent then informs the broker 14 to accept the TCP connection requested by the client 10. The broker 14 then releases the original TCP SYN packet, where the system employs a conventional transparent proxy to terminate the TCP connection at the broker. All data from the client 10 is passed to the agent 16 through the agent-established data session, where the agent 16

passes the data to the other side of the proxy, towards the server. A similar process operates in the reverse path.

If the connection fails, then the agent 16 informs the broker 14 as to the type of failure. The broker consumes the original TCP SYN packet and generates a new IP packet to be sent to the client 10. The new packet 10 is so constructed that it is a replica of the condition detected by the agent 16: that is, TCP RST ACK, or an ICMP packet. Correct information relating to the connection failure is therefore returned to the client 10. In the case of a TCP RST ACK packet, the packet is addressed so that it appears to the client 10 to have originated from the destination host. In the case of an ICMP packet, the packet is addressed so that it appears to the client 10 to have originated from the gateway address of the broker 14 and carries the first 64 bytes of the original TCP SYN packet, as required by the ICMP protocol defined in RFC 792. In this manner, the brokered gateway is completely transparent to the client 10 at TCP level.

The method used by the broker 14 and agent 16 to establish a transparent connection at the network layer for Internet Control Message Protocol (ICMP) between the client 10 and the server 12 will now be described.

ICMP is a mandatory requirement for gateway devices. The embodiment supports ICMP by consuming ICMP packets, interpreting them at the application layer, and if necessary creating an application layer protocol message exchange from the broker 14 to an agent 16.

For example, assume that the client 10 wishes to ping the server 12. It does this by sending an ICMP echo request message, with payload, from the client 10 to the server 12. The ICMP message is consumed by the broker 14, where it is parsed to determine its type. In this case, it will be identified as a PING request. The destination address (the server 12) is looked up and a routing decision is made to generate a PING message to the agent 16, over the secure transport layer protocol. The agent 16, on receipt of the PING message completes the request by generating an ICMP echo request message from itself to the server 12. On receipt of an ICMP echo reply, the agent 16 responds with an "OK" response to the PING. At the broker 14, the application layer response to the PING message determines the ICMP response to the original ICMP request message: ICMP echo reply (server 12 to client 10); or ICMP unreachable (broker 12 to

client 10). An ICMP echo request to the gateway address is handled by the broker 14, without the need to route to an agent 16. Other ICMP messages are handled in a similar fashion, so making the embodiment ICMP transparent.

The method used by the broker 14 and agent 16 to establish a transparent connection at the network layer for the User Datagram Protocol (UDP) between the client 10 and the server 12 will now be described.

UDP, the User Datagram Protocol, is a very simple protocol that allows a packet of data, known as a datagram, to be sent from one host to another. The embodiment supports UDP by consuming defragmented UDP packets at the broker 14, and routing appropriately to agents 16.

Assume that the client 10 sends a UDP datagram to the server 12. The broker 14 consumes the datagram and relays the datagram to the agent 16, from where the datagram is relayed to the server 12, having the address of the agent 16 as its source address and an agent-selected source relay port. Datagram replies follow the reverse path from the server 12 to the agent-selected relay port on the agent 16. The agent 16 receives the reply and transports the datagram to the broker 14 using the secure transport. The datagram reply is sent to the client 10 having as its source address that of the server 12.

When a broker identifies which agent to which a datagram should be routed, it requests the establishment of a datagram relay. To establish a datagram relay, an agent first establishes a named datagram session between itself and the serving broker. The agent, if it accepts the relay request, responds with the name of the datagram session allocated to the datagram relay. The broker then sends the datagram, as a full IP packet, over the named datagram session, and stores the relay details (protocol = UDP, source address, source port, destination address, destination port). The agent sends the datagram to the final destination server using the established datagram relay.

An established datagram relay supports replies in the return direction, where the agent receives the reply and sends the datagram, as a complete IP packet, over the named datagram session. The broker is responsible for packet fragmentation and raw transmission of the datagram to the host.

The datagram session between broker and agent is a bi-direction secure pipe. The pipe is reliable. Datagram transports however should by definition be unreliable. Therefore, within the embodiment, strategies are employed on the datagram sessions to mimic unreliability. A flow-control protocol is implemented over the datagram session to provide end-to-end information on the number of outstanding bytes between the datagram session peers. If, for any reason, there is congestion in the datagram sessions, the peers will drop datagrams. If an underlying datagram session disconnects, pending datagrams are dropped. Other known traffic management strategies can be implemented at the broker and agent. For example, the embodiment may implement "ageing" of datagrams: that is, datagrams are dropped if the length of time they have been queued exceeds a configurable threshold.

While the embodiment described above supports UDP, the principle of operation is easily extensible to any datagram-based service.

The embodiment supports the ability for multiple virtual networks to be serviced by a brokered network gateway through a client access network. The client access network is responsible for managing client access, for example through a combination of standard AAA (Authorisation, Authentication and Accounting) services such as RADIUS, and VPN services such as IPSec or MPLS.

Each virtual network consists of a number of clients 10, agents 14 and a virtual gateway IP address. The broker restricts client access to within the client's virtual network. The client's identity is determined by IP source address (non-overlapping client-side IP address range). Each virtual network operates on an overlapped server-side IP address range as communication between clients and servers are within the context of the virtual network. Client trust relationships to a virtual network can be established dynamically through existing AAA services, using either static or dynamic IP address assignment.

The structure of the broker 14 will now be described with reference to Figure 2.

The broker 14 is currently implemented on a server computer running the Linux operating system.

Agents 16 establish and maintain a control session over a secure transport (TLS over TCP, in this embodiment) and a number of dynamic data sessions. A secure transport

module 30 is responsible for providing a secure transport, interacting with the TCP module 32 of the Linux kernel, which in turn receives and transmits IP packets via "IP INPUT" and "IP OUTPUT" respectively.

On a serving broker, control sessions are maintained by a control module 34. Data sessions are of two forms, STREAM sessions and DGRAM sessions. On an operational broker, STREAM sessions are maintained by a STREAM manager module 36 and DGRAM sessions are maintained by a DGRAM manager module 38. Agents are responsible for the establishment of all sessions, therefore enabling traversal of intervening firewalls and proxies. Data sessions can be established on demand or pre-connected.

All IP traffic passes through the kernel; inward IP traffic passes through an IP INPUT module 40, where it is passed to an IP Filter (PREROUTING) module 42. Using iptables, the Linux kernel packet filtering module, rules are established to queue packets to the application layer module called Packet DeMux 44.

The module Packet DeMux 44, based on information maintained by a serving broker database module 46 is used to determine how to process each packet passing though the broker, based on the IP protocol, the originating client, destination server and depending on protocol the source and destination protocol ports.

The Packet DeMux module 44 also implements an application-layer, configurable packet filter (firewall) based on Access Control List (ACL) rules stored in the serving broker database 46. The present embodiment supports per-client defined ACLs and per-network ACLs.

When the broker is handling TCP packets, only SYN packets are queued on the input of the Packet DeMux module 44. If a TCP SYN packet represents a valid new TCP connection for a valid client (as defined by the contents of the serving broker database 46), a SYN message, with the details of the packet, are passed to the control module 34. The control module 34 locates a valid agent control session, and offers the new TCP connection through the established control session. If no agent route is found, then the control module 34 generates a raw ICMP Unreachable packet for transmission to the originator of the TCP request through an IP output module 48.

If an agent's response to a new TCP connection is "OK", then the assigned STREAM session is located in the serving broker database 44 and set to be in a "parked" state, saving the TCP connection details. A message is then sent to the Packet DeMux module 44 to instruct it to respond to the original TCP SYN packet with an "ACCEPT", allowing the TCP SYN packet to reach the kernel TCP module 32. The IP Filter (PREROUTING) module 42 module is configured to operate as a transparently proxy, using destination NAT, to a transparent proxy port, on which the stream manager module 36 listens. The stream manager module 36 accepts the new TCP connection and asks the IP Filter (PREROUTING) 42 module what the original destination address of the connection was. Based on the source and the destination addresses and ports, the parked STREAM session is located and unparked. The established proxy is sent to a stream proxy 50 to manage until the connection is finished.

If the agent's response to a new TCP connection is "REFUSE", then the control module 34 sends a message to the Packet DeMux module 44 to refuse the TCP connection. In this case, the original TCP SYN packet is modified into a TCP RST packet and the source and destination are swapped to reflect the packet to the client. The original packet is then consumed (DROP) and the modified packet is sent to the IP OUTPUT module 48.

If the agent's response to a new TCP connection is "UNREACHABLE", then the control module 34 sends a message to the Packet DeMux module 44 to respond to the TCP SYN with an ICMP UNREACH packet. In this case the original TCP SYN packet is modified into an ICMP UNREACH packet and the source and destination are swapped to reflect the packet to the client. The first 64 bytes of the original TCP SYN packet are copied into the modified ICMP UNREACH packet according to the ICMP specification. The original packet is then consumed (DROP) and the modified packet is sent to the IP OUTPUT module 48.

Handling of ICMP will now be described. ICMP packets are queued by the IP Filter (PREROUTING) module 42 to the Packet DeMux module 44. In the case of an ICMP Echo Request, a "PING" message is sent to the control module 34, where a "PING" message is sent, through an established CONTROL session to a selected (routed) agent, based on information in the serving broker database 46. The original ICMP Echo Request is dropped.

Depending on the agent's response to the PING, a new ICMP message is created and sent to the client as a raw IP packet through the IP OUTPUT module 48.

Handling of UDP will now be described. UDP packets are queued by the IP Filter (PREROUTING) module 42 to the Packet DeMux module 44. The "IP Filter (PREROUTING) module 44 defragments the UDP packets. However, an alternative implementation could allow defragmentation to occur in the Packet DeMux module 44. The information in the IP and UDP headers of the UDP datagram are used to select/lookup an established datagram association (DA) in the serving broker database 46. If a DA is found, then information in the DA determines either that the datagram should be discarded, or forwarded directly to an assigned DGRAM RELAY 52. If no DA is found, then a new DA is created, and the datagram packet is forward to the control module 34. The control module 34, using information in the serving broker database 46 determines which agent to request the establishment of a new datagram relay.

If the agent responds with "OK", then the DGRAM session assigned is located in the serving broker database 46 and the DA is bound a DGRAM RELAY 52. The original packet is then forwarded to the DGRAM RELAY 52.

If the agent responds with "REFUSE", then the datagram packet is discarded and the DA is marked "discard". The "discard" condition is timer managed, causing the DA to be automatically destroyed after a configurable period of time.

The DGRAM RELAY 52 is responsible for managing traffic over DGRAM sessions. Each DGRAM RELAY 52 implements flow-control, DA maintenance, and traffic management strategies. Any IP datagram can be transported over a DGRAM session. Datagrams in the reverse path (agent to client) are received by a DGRAM RELAY 52 and transmitted, after fragmentation, via the IP OUTPUT module 48.

Virtual networks are assigned to one of a number of deployed serving brokers. Agents can use a registration service to discover the serving broker assigned as a gateway to the agent's virtual network. In this way an embodiment of the invention can be scaled to support a large number of clients, agents and virtual networks. In the preferred embodiment, the registration service is implemented using secure web-service (using HTTPS methods).

The agent can be implemented on any platform that provides a TCP/IP network stack, for example, a computer running Windows (rtm), Linux (rtm), Unix (rtm), Apple (rtm) Operating System or a Java (rtm) virtual machine. The generic structure of an agent is shown in the Figure 3. The platform must provide interfaces to TCP/IP. In Figure 3, TCP interface 60, UDP interface 62 and ICMP interface 64 of the native TCP/IP stack are shown separately for clarity.

All traffic between the agent 16 and serving broker 14 is established as outbound (agent to serving broker) secure connections using the SECURE TRANSPORT module 66, which provides TLS over TCP in the preferred embodiment. Since agents are responsible for traversing intervening proxies between the agent 16 and serving broker 14, an agent must include proxy connection functionality, the functionality for which can be included in the SECURE TRANSPORT module 66, or as an alternative be included in a separate proxy module. An agent should implement HTTP CONNECT proxy functionality as a minimum.

On an agent 16, control sessions are maintained by a CONTROL module 68. Data sessions are of two forms, STREAM sessions, managed by a STREAM MANAGER module 70, and DGARM sessions, managed by a DGRAM MANAGER 72.

The CONTROL module 68 of the agent 16 establishes and maintains a single secure control session via the SECURE TRANPORT module 66. The CONTROL module 68 listens for control messages from the broker 14 over the control session. The CONTROL module 68 should be capable of handling multiple serving broker messages concurrently.

If a serving broker 14 sends a PING message, then the CONTROL module 68 will initiate an ICMP ping from the agent 16 to the destination through the TCP/IP ICMP interface 60. Someone proficient in the technical field will realise that an ICMP ping is typically only allowed by processes with elevated or administrator rights. As a result, an agent 16 must typically run in a context with elevated or administrator rights. The response to the ICMP will be related back to the serving broker 14 in a response message.

If a serving broker 14 sends a TCP CONNECT message to the agent 16, then the CONTROL module 68 requests a TCP connection by way of the STREAM manager

module 70. The STREAM MANAGER module 70 must first attempt to connect to the specified destination. If the connection succeeds, then the STREAM MANAGER module 70 will connect a STREAM session to the serving broker 14 and notify the CONTROL module 68 of the success and the STREAM session that is responsible for the TCP connection. The STREAM MANAGER module 70 then creates a STREAM PROXY 74 to shunt data back-and-forth between the STREAM session and local TCP connection 78. If however the TCP connection fails, then the STREAM MANAGER module 70 notifies the CONTROL module 68, which sends the appropriate error response message. Alternatively, an agent 16 STREAM MANAGER module 70 can pre-connect STREAM sessions to the serving broker 16 and allocate pre-connected STREAM sessions when needed, in this manner TCP connection establishment times can be improved.

If a serving broker 14 sends a UDP CONNECT message to the agent 16 then the CONTROL module 68 requests a UDP Datagram Association (DA) via the DGRAM manager 72. The DGRAM MANAGER 72 assigns the DA to a DGRAM RELAY 76. If the DGRAM RELAY 76 does not already have a DGRAM session to the serving broker 14, then it is established now. A DGRAM RELAY 76 is capable of managing multiple DAs. The DGRAM MANAGER 72 sends a response message to the serving broker 14 identifying the DGRAM session to which to bind the DA of the serving broker 14.

The DGRAM RELAY 76 of the agent 16 preferably implements a timer function to monitor DAs that are alive and unbind DAs that are no longer being used. When a DA is unbound, the DGRAM RELAY 76 of the agent 16 sends the DGRAM RELAY 52 of the serving broker 14 a message to that effect. After the DA has been unbound, any traffic for the unbound DA is discarded.

In practice, most UDP protocols are one-shot command-response style protocols. With this in mind, an agent could implement a very short idle time-out initially, for example 30 seconds, and then if there are multiple datagrams exchanged, extend the idle timeout. For example, after two response datagrams, the agent 16 may extend the idle timeout for that DA to 15 minutes.

Both brokers 16 and agents 14 are, in this embodiment, identified and mutually authenticated using X.509 digital certificates. Routing in the system may be based on a destination address or network service.

Claims

1. A data transmission system for secure data exchange using transmission control protocol between a client and a server, comprising an agent and a broker connected to exchange data over an unsecured network link, in which:

upon receipt of a control packet from the client, the broker forwards a modified control packet to the agent using a secure protocol;

the broker inspects the modified control packet and forwards it to the server;

upon receipt of a response packet from the server, the agent forwards the response packet to the broker using a secure protocol; and

upon receipt of the response packet, the broker modifies the response packet and forwards it to the client;

wherein in the case that the exchange of control packets indicates establishment of a TCP session, the broker and the agent establish a data channel between themselves to create a transparent TCP channel between the client and the server.

2. A data transmission system according to claim 1 in which the client is connected to the broker for exchange of data over a secured network.

3. A data transmission system according to claim 2 in which the secured network is connected to the unsecured network by one or more of a firewall, a proxy or a network address translation (NAT) device.

4. A data transmission system according to any preceding claim in which the secure protocol is a transport layer security (TLS) protocol.

5. A data transmission system according to claim 4 in which the TLS connection is made over the TCP port 443.

6. A data transmission system according to claim 4 in which the TLS connection is made over a local proxy.

7. A data transmission system according to claim 6 in which the local proxy uses the HTTP CONNECT protocol.

8. A data transmission system according to any preceding claim in which the control packet is a TCP SYN packet or a TCP ACK packet.

9. A data transmission system according to any preceding claim in which the response packet is a TCP SYN ACK or a TCP RST ACK packet.

10. A data transmission system according to any preceding claim in which the control packet and the response packet are Internet control message protocol packets.

11. A data transmission system according to any preceding claim in which the broker applies rules to determine whether or not the control packet is forwarded to the agent.

12. A data transmission system according to claim 11 in which the rules specify whether a the control packet should be forwarded based upon one or more of the originator of the packet, the target address of the packet, and the target port number of the packet.

13. A data transmission system according to any preceding claim operative to service multiple virtual networks through a client access network.

14. A data transmission system according to claim 13 in which the client access network is responsible for managing client access.

15. A data transmission system according to claim 14 in which the client access network uses one or both of standard AAA (Authorisation, Authentication and Accounting) and VPN services.

16. A data transmission method for secure data exchange using transmission control protocol between a client and a server, using an agent and a broker connected to exchange data over an unsecured network link, in which:

upon receipt of a control packet from the client, the agent forwards a modified control packet to the broker using a secure protocol;

the broker inspects the modified control packet and forwards it to the server;

upon receipt of a response packet from the server, the broker forwards the response packet to the agent using a secure protocol; and

upon receipt of the response packet, the agent modifies the response packet and forwards it to the client;

wherein in the case that the exchange of control packets indicates establishment of a TCP session, the agent and the broker establish a data channel between themselves to create a transparent TCP channel between the client and the server.

17. A data transmission method according to claim 16 in which the client is connected to the broker for exchange of data over a secured network.

18. A data transmission method according to claim 17 in which the secured network is connected to the unsecured network by one or more of a firewall, a proxy or a network address translation (NAT) device.

19. A data transmission method according to any one of claims 16 to 18 in which the secure protocol is a transport layer security (TLS) protocol.

20. A data transmission method according to claim 19 in which the TLS connection is made over the TCP port 443.

21. A data transmission method according to claim 20 in which the TLS connection is made over a local proxy.

22. A data transmission method according to claim 21 in which the local proxy uses the HTTP CONNECT protocol.

23. A data transmission method according to any one of claims 16 to 22 in which the control packet is a TCP SYN packet or a TCP ACK packet.

24. A data transmission method according to any one of claims 16 to 23 in which the response packet is a TCP SYN ACK or a TCP RST ACK packet.

25. A data transmission method according to any one of claims 16 to 24 in which the control packet and the response packet are Internet control message protocol packets.

26. A data transmission method according to any one of claims 16 to 25 in which the broker applies rules to determine whether or not the control packet is forwarded to the agent.

27. A data transmission method according to claim 26 in which the rules specify whether a the control packet should be forwarded based upon one or more of the originator of the packet, the target address of the packet, and the target port number of the packet.

28. A data transmission method according to any one of claims 16 to 27 operative to service multiple virtual networks through a client access network.

29. A data transmission method according to claim 28 in which the client access network is responsible for managing client access.

30. A data transmission method according to claim 29 in which the client access network uses one or both of standard AAA (Authorisation, Authentication and Accounting) and VPN (virtual private networking) services.

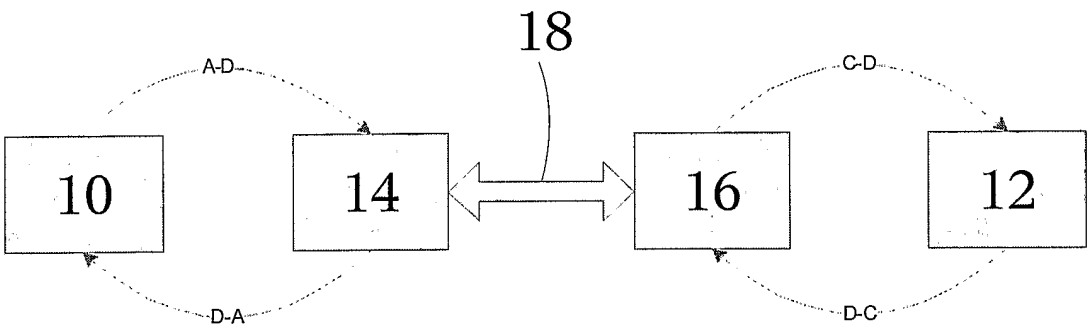


Fig 1

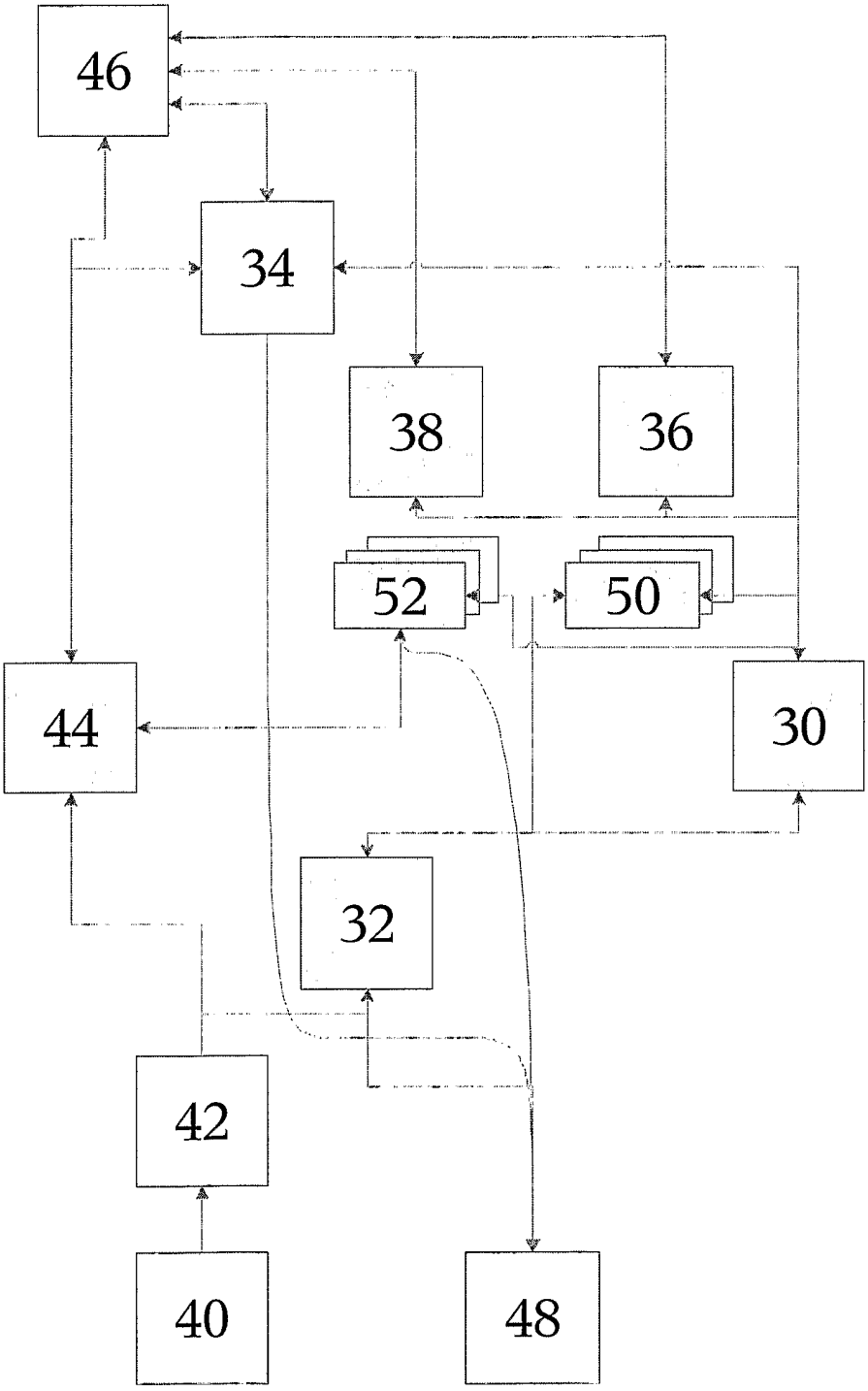


Fig 2

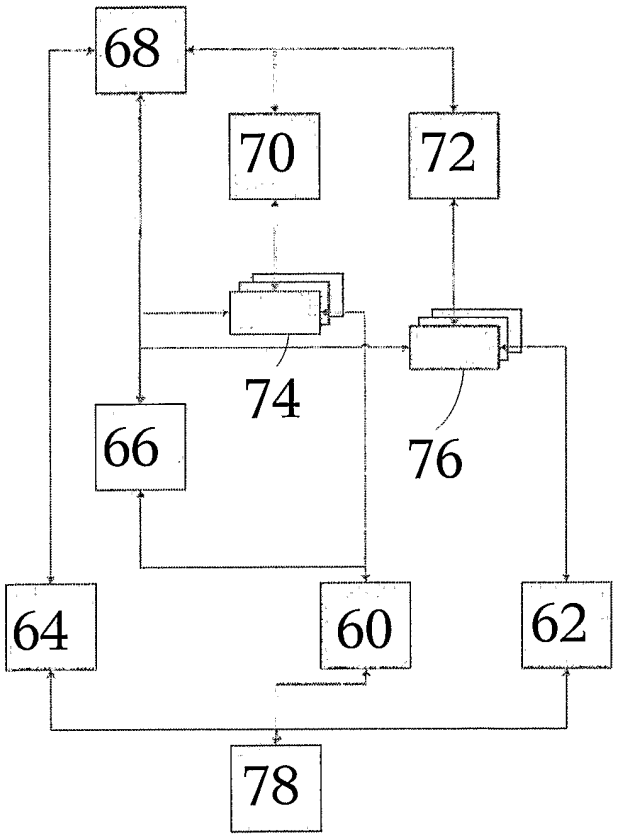


Fig 3

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2006/001681

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L

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

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2002/042875 A1 (SHUKLA JAYANT) 11 April 2002 (2002-04-11) paragraph [0074] - paragraph [0080] figures 2,5,9	1-30

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
12 May 2006	24/05/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Horn, M.P.

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2006/001681

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2002042875 A1	11-04-2002	NONE	