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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 July 2015 (16.07.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/104567 A1

(51) International Patent Classification:
G06F 21/62 (2013.01) *H04L 9/30* (2006.01)

(21) International Application Number:
PCT/HU2014/000007

(22) International Filing Date:
13 January 2014 (13.01.2014)

(25) Filing Language: English

(26) Publication Language: English

(72) Inventor; and
(71) Applicant : **BALAZS, István József** [HU/HU]; Gyömröi Aurél u. 3. Fszt.3., H-2230 Gyömrö, Hungary (HU).

(74) Agent: **KACSUKPATENT KFT.**; Zsófia KACSUK, Üteg utca 11/A, H-1139 Budapest (HU).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) **Title**: SECURE COMMUNICATION BETWEEN A SERVER AND A CLIENT WEB BROWSER

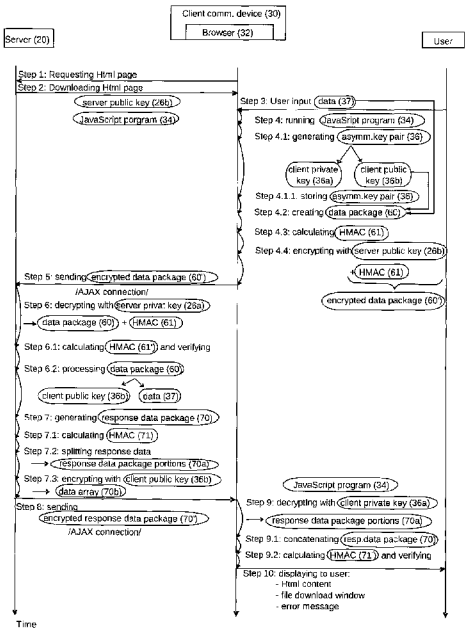


Fig. 2

(57) **Abstract**: The invention relates to a method for allowing secure communication between a server and a client web browser running on a client communication device which is connectable with the server over an electronic communication channel, characterised by comprising: - providing a HTML webpage at the server - sending a JavaScript program to the client web browser when the HTML webpage is downloaded by the client web browser, which JavaScript program is configured, when executed by the client web browser, to: generate a client asymmetric key pair consisting of a client public key and a client private key, create a first client data package comprising the client public key, and send the first client data package to the server over the electronic communication channel.

WO 2015/104567 A1

SECURE COMMUNICATION BETWEEN A SERVER AND A CLIENT WEB BROWSER

The present invention relates to method for performing secure communication between a server and a client web browser running on a client communication device which is connectable with the server over an electronic communication channel. The secure communication according to the invention can be used for example for authentication purposes or credit card payments or for sending encrypted e-mails or files.

A PKI (public key infrastructure) enables users of a basically unsecure public network such as the Internet to securely and privately exchange data through the use of a private and a public cryptographic key pair which is generally obtained and shared through a trusted authority. The public and a private cryptographic key pair is often referred to as asymmetric key pair.

In the asymmetric key cryptography, a public and private key are created simultaneously using the same algorithm (a popular one is known as RSA) by a certificate authority (CA). The private key is given only to the requesting party and the public key is made publicly available generally as part of a digital certificate in a directory that all parties can access. The private key is kept secret and never shared with anyone or sent across the Internet. A third party wishing to send encrypted data to the holder of the private key, obtains the public key (e.g. by downloading it from a public directory or by receiving it from the holder of the private key, or in any other way). The third party encrypts the data with the obtained public key and sends the encrypted data to the holder of the private key over an electronic communication channel such as the internet. The private key is used by its holder to decrypt the encrypted data.

The PKI key pairs are also used for authenticating the holder of the private key. In this case the holder of the private key encrypts a digital certificate with the private key. The person authenticating the holder uses the public key to decrypt the digital certificate. Since the private key is kept secret only the holder of the private key is able to create the encrypted digital certificate.

A public key infrastructure is a rather complex system consisting of a certificate authority (CA) that issues and verifies digital certificates; a registration authority (RA) that acts as the verifier for the certificate authority before a digital certificate is issued to a requestor; one or more directories where the certificates are held with their public keys; and a certificate management system. Use of the public key infrastructure is also somewhat inconvenient as the person requesting a private key is generally required to personally visit the registration authority for verifying his or her identity and to personally collect the private key as it would be insecure to transmit the private key over the Internet.

The public key infrastructure assumes the use of public key cryptography, which is the most common method on the Internet for authenticating a message sender or encrypting a message. Traditional cryptography has usually involved the creation and sharing of a secret key for the encryption and decryption of messages. This secret key system is often referred to as symmetric cryptography and the secret key as symmetric key. The advantage of symmetric cryptography is that smaller sized symmetric key is sufficient to encrypt and decrypt larger data as compared to the required key size in case of asymmetric cryptography. However, symmetric cryptography has the significant flaw that the symmetric key has to be shared between the two parties wishing to exchange data and if the symmetric key is intercepted by someone else, messages can easily be decrypted. For this reason, public key cryptography is the preferred approach on the Internet.

It is an object of the present invention to overcome the problems associated with the prior art. In particular, it is an object of the invention to provide a more simple and convenient way of sending encrypted messages over the Internet and authenticating a party visiting a website on the Internet. It is a further object of the invention to provide these in a manner which does not require additional software installation on the client side, in particular it is an object to realise the preceding objects via JavaScript and HTML 5.

These objectives are achieved by a method for allowing secure communication between a server and a client web browser running on a client communication device which is connectable with the server over an electronic communication channel, characterised by comprising:

- providing a HTML webpage at the server

- sending a JavaScript program to the client web browser when the HTML webpage is downloaded by the client web browser, which JavaScript program is configured, when executed by the client web browser, to:

  - generate a client asymmetric key pair consisting of a client public key and a client private key,
  - create a first client data package comprising the client public key, and
  - send the first client data package to the server over the electronic communication channel.

Providing (sending) the JavaScript program which is configure, when executed by the client web browser, to perform the above (and further) steps is understood to describe a method, wherein the given steps are performed via (with the help of) the JavaScript program. In the present case the method according to the invention comprises the steps of:

- generating with the help of the JavaScript program a client asymmetric key pair consisting of a client public key and a client private key,

- creating with the help of the JavaScript program a first client data package comprising the client public key, and

- sending with the help of the JavaScript program the first client data package to the server over the electronic communication channel.

Further advantageous embodiments of the invention are defined in the attached dependent claims.

Further details of the invention will be apparent from the accompanying figures and exemplary embodiments.

Fig. 1 is a schematic block diagram of an exemplary system for performing the method according to the invention.

Fig. 2 is a schematic flow diagram illustrating a preferred embodiment of the method according to the invention.

Fig. 2a is schematic flow diagram illustrating an alternative of the embodiment depicted in Fig. 2.

Fig. 3 is a schematic flow diagram illustrating another preferred embodiment of the method according to the invention.

Fig. 4 is a schematic flow diagram illustrating another preferred embodiment of the method according to the invention.

Fig. 5 is a schematic flow diagram illustrating another preferred embodiment of the method according to the invention.

Fig. 1 schematically illustrates an exemplary system 10 for carrying out the method according to the invention. The system 10 comprises a server 20 and a client communication device 30 which is connectable with the server 20 over the Internet 40 over an electronic communication channel 50 the latter being a HTTP or HTTPS channel, that is known to the skilled person. The server 10 is a web server capable of HTTP(S) based communication and hosting a plurality of HTML web pages 22a belonging to one or more HTML websites 22. The 10 server stores a private key 26a and a public key 26b of an asymmetric key pair 26.

The client communication device 30 may be any kind of communication device that is suitable for running a web browser 32 that is capable of HTTP(S) based communication with the web server 20 and is capable of running Javascript code. For example the client communication device 30 may be a personal computer, laptop, notebook, tablet, smart phone, etc. The web browser 32 can access the web page 22 hosted by the web server 20 over the Internet 40 by establishing the electronic communication channel 50 between the server 20 and the communication device 30. The electronic communication channel 50 corresponds to the type of communication device 30 used, and it may be any kind of conventional communication channel, such as cable or wireless telecommunication channel, in particular Internet network, mobile phone network, such GSM network, CDMA network, Wifi network, etc. The electronic communication channel 50 uses the HTTP protocol, which is the seventh application layer of the ISO/OSI layers.

In the course of the procedure according to the invention the browser 32 running on the client communication device 30 downloads a JavaScript program 34 from the Server 20 over the electronic communication channel together with the HTML web page 22 that is being visited. The JavaScript program 34 is configured,

when executed, to generate an asymmetric key pair 36 comprising a private key 36a and a public key 36b, which are either stored in a local storage 38 of the web browser 32 or in the memory area 39 of the communication device 30 which is being used by the web browser 32, depending on the type of procedure as will be explained later on.

Fig. 2 illustrates the steps of a preferred embodiment of the method according to the invention which is used to encrypt messages.

In step 1 the web browser 32 running on the client communication device 30 connects to the server 20 over the Internet 40 by establishing the electronic communication channel 50 and requests a HTML page 22. This step is generally preceded by opening an URL in the browser 32 either as a result of data input from a user of the client communication device 30 (e.g. the user types the URL into the address bar of the browser 32) or a command executed by the browser or another program running on the client communication device 30. The electronic communication channel 50 opened by the browser 32 of the client communication device 30 is preferably a HTTP(S) (Hyper Text Transfer Protocol (Secure)) channel.

In step 2 the HTML page 22 is downloaded from the Server 20, whereby the JavaScript program 34 is sent together with the code of the HTML page 22. Optionally the server public key 26b may be sent/downloaded too, or alternatively, the server public key 26b may already have been downloaded from a public directory or made accessible to the browser 32 and later on for the JavaScript program 34 in any other suitable way. If the server public key 26b is sent in step 2, it may preferably be stored in a local storage 38 of the client web browser 32. The HTML 5 specification allows the web browser 32 to permanently store data in its local storage 38. Data stored in the local storage 38 is only accessible by a JavaScript program 34 running on the given domain, in the present case any JavaScript program 34 downloaded from the first visited HTML webpage 22a or any other HTML webpage belonging to the HTML website 22 of the first visited webpage 22a (thus belonging to the same domain). Data stored in the local storage 38 does not expire and is kept even after restarting the web browser 32 or the client communication device 30. It should be further noted that the local storage 38 is part of all web browsers 32 that are capable of HTML 5 (including mobile phone web browsers 32). It should be appreciated that the server public

key 26b may also be stored in the memory 39 of the client communication device 30.

Further data may also be sent in step 2 to the client web browser 32 such as the HTTP session identifier(s).

After the HTML webpage 22a is downloaded and displayed to the user of the client communication device 30, the user is generally prompted to input data 37 (e.g. into the input fields of the HTML webpage 22a, including file upload) or a command (e.g. press a command button, such as a "get mail" or "download mail" command button). After the client web browser 32 receives the user input in step 3, the browser 32 launches the JavaScript program 34 in step 4.

The JavaScript program 34 generates in step 4.1 the asymmetric client key pair 36 consisting of the client private key 36a and the client public key 36b. The format of the client asymmetric key pair 36 preferably correspond to the openssl library format (PEM format: Base64 and ASN.1 coding). The size of the keys 36a, 36b as well as the exponent used to generate the key pair 36 is preferably variable, and it is preferably chosen by the JavaScript program 34 in accordance with the size of the data to be encrypted, the desired strength of the encryption and the estimated processor capacity. The latter may be estimated for example by repeatedly running a test cycle (e.g. for a million times) and by measuring the run time. Having regard to the presently common processor speed and memory size, the generation of a key pair 36 consisting of keys 36a, 36b, wherein the private key 36a has a size of 512 bytes, takes about 4 to 6 seconds. In order to increase the security level the key pair 36 is only used for a single client-server-client request, and a new key pair 36 is generated for a next client request.

The asymmetric key pair 36 may be stored temporarily or permanently in step 4.1.1 depending on the application. When the method according to the invention is applied for sending encrypted messages preferably an AJAX (Asynchronous JavaScript and XML) connection is used as the communication channel 50 between the server 20 and the client communication device 30, which is kept open during the communication session and does not require reloading the whole of the HTML webpage 22a. In this case the asymmetric key pair 36 may be stored temporarily in the memory 39 of the client communication device 30, preferably in the form of a variable of the JavaScript program 34.

In step 4.2 the JavaScript program 34 creates a first client data package 60 comprising the client public key 36b and optionally other data 37, such as any data 37 inputted (typed) by the user in step 3. The method preferably includes step 4.3 in which the JavaScript program calculates a HMAC 61 (hash-based message authentication code) of the first client data package 60, which later on allows the server 20 to tamper proof the client data package 60, i.e. to determine whether or not the first client data package 60 has been tampered. In step 4.4 the JavaScript program 34 encrypts the first client data package 60 with the server public key 26b and concatenates it with the HMAC 61 whereby an encrypted first client data package 60' (containing the HMAC 61 as well) is obtained, which is then sent to the server 20 over the electronic communication channel 50 (preferably an AJAX connection) between the server 20 and the client communication device 30 in step 5.

Optionally, if the obtained data package 60 is larger than the maximum encryptable data size determined by the size of the server public key 26b (being e.g. 2048 bytes) then the method further includes the step of splitting the first client data package 60 into data package portions 60a having sizes that can be encrypted by the server public key 26b (step 4.31 illustrated in Fig 2a). After this, in step 4.5, the JavaScript program 34 encrypts the first client data package 60 by encrypting each data package portion 60a with the server public key 26b and creates an encrypted data package 60' in the form of a data array 60b of the encrypted data package portions 60a that is sent to the server 20 in step 5 over the AJAX communication channel 50 between the server 20 and the client communication device 30 using HTTP GET command. The HMAC 61 of the original data package 60 can be calculated in this case as well, which is concatenated with the data array 60b in order to create the encrypted data package 60' that is sent to the server 20 in step 5. The step 4.3 of calculating the HMAC precedes the step 4.31 of splitting up the client data package 60.

The encrypted data package 60' is decrypted by the server 20 in step 6 using the server private key 26a. If the original client data package 60 has been split into data package portions 60a, these are decrypted separately and concatenated after decryption in order to restore the client data package 60. In case the encrypted data package 60' contained the HMAC 61 as well, then in step 6.1 the server 20 calculates a new HMAC 61' for the first client data package 60

and- compares the calculated new HMAC 61' with the extracted received HMAC 61 for tamper proofing the client data package 60. If the newly calculated HMAC 61' is identical to the received HMAC 61 this means that the client data package 60 has not been tampered with. If the new HMAC 61' is not identical to the received HMAC 61 this indicates that the client data package 60 has been tampered with and in this case the server preferably aborts the process (e.g. by navigating the client browser to an error message URL or by braking up the established communication channel 50).

If the tamper proofing is negative (i.e. there is no indication of tampering) then the server 20 proceeds with processing the received client data package 60 in step 6.2. This step includes extracting the client public key 36b and other data 37 from the client data package 60.

In step 7 the server generates response data package 70, which may comprise for example HTML data for reloading part of the HTML webpage 22a (e.g. for displaying a new HTML form), a file or an error message (e.g. incorrect or missing data inputted in an input field of the HTML webpage 22a). A HMAC 71 may optionally be calculated in step 7.1 for the response data package 70.

The client public key 36b is preferably of a smaller size, e.g. 512 bytes, in order not to require long time for generating the client key pair 36. Accordingly, the response data package 70 generally needs to be split into smaller response data package portions 70a determined by the size of the client public key 36ab in step 7.2 as explained in connection with the first client data package 60 of the embodiment depicted in Fig. 2a. The data package portions 70a are then encrypted separately in step 7.3 and a data array 70b is created from the encrypted data package portions. If the HMAC 71 of the original response data package 70 has been calculated this is preferably concatenated with the data array 70b whereby an encrypted response data package 70' is obtained. If no HMAC 71 has been calculated then the encrypted response data package is the encrypted data array 70b.

The encrypted response data package 70' is sent to the client web browser 32 in step 8 over the open AJAX communication channel 50, preferably using JSON coding, which allows for preserving and restoring the data structure created by the server 20.

The client web browser 32 receives the encrypted response data package 70' and the data package portions 70a of the data array 70b are each decrypted in step 9 by the JavaScript program 34 using the client private key 36a. The HMAC 71 of the original data package 70 is also extracted from the encrypted response data package 70'. The data package portions 70a are then concatenated in step 9.1 in order to obtain the response data package 70. If a HMAC 71 has been used by the server 20, in step 9.1 the JavaScript program 34 verifies the HMAC 71 by calculating a new HMAC 71' from the received response data package 70 and by comparing the calculated HMAC 71' with the extracted HMAC 71 in order to determine whether or not the response data package 70 has been tampered with. If the tamper proofing is negative, i.e. the response data package has not been tampered with, then information corresponding to the response data package 70 is displayed to the user in step 10. For example new HTML content is displayed, or a file download window is displayed, or an error message is displayed, etc.

File downloading is achieved by way of a special HTTP message. Although the file download window is displayed to the user, and the file is downloaded that has been sent by the server 20 in the response data package 70. It should be noted that this file downloading is only virtual since the file has already been transmitted to the client communication device 30 via AJAX response.

Having regard to the processor speed and the memory of commonly available client communication devices 30, the transmission of big files (larger than 500 KBytes) would require very long time. In order to overcome this problem a preferred embodiment of the method according to the invention uses symmetric cryptography for the transfer of big files as illustrated in Fig. 3.

After the file download window is displayed to the user in step 10, the user may select the file download option in step 11 in any known way (e.g. pressing a corresponding control button on the file download window). According to a possible embodiment a big file 72 is not sent to the client communication device 30 before the user chooses to download it. If the user does choose to download the big file 72 in step 11, then the file download instructions can be sent to the server 20 by repeating steps 4.1 to 6.2.

After this a new response data package 70 is generated in step 17 which includes or corresponds to the big file 72. A symmetric key 74 is generated in step 17.1 by the server 20. This step may precede any of the prior steps and the

symmetric key 74 may already be stored on the server 20 by this time. The file 72 is then encrypted with the symmetric key 74 in step 17.2, whereby an encrypted file 72' is obtained. The symmetric key 74 is also encrypted using the client public key 36b in step 17.3 thus obtaining an encrypted symmetric key 74'. In step 18.1 the encrypted symmetric key 74' is sent to the client web browser 32 over the AJAX communication channel 50. The encrypted file 72' is also sent to the client web browser 32 over the AJAX communication channel 50 in step 18.2. It should be noted that the encrypted symmetric key 74' and the encrypted file 72' may be sent as part of a single data package in a single step. Furthermore, the transmission of the encrypted file 72' may precede the transmission of the encrypted symmetric key 74.

It should be further noted that a HMAC may be calculated for the original file 72 and sent together with any one of the response messages, to be used for tamper proofing by the JavaScript program 34 as discussed above.

In step 19 the JavaScript program 34 decrypts the encrypted symmetric key 74' thus obtaining the symmetric key 74. In step 20 the JavaScript program uses this symmetric key 74 to decrypt the encrypted file 72' in order to obtain the original big file 72. In step 21 this file is provided to the user (e.g. opened and displayed or save on the client communication device 30).

It should be noted that steps 17 to step 20 may also be incorporated into steps 7 to 9, whereby the big file 72 is already available on the client communication device 30 when the user selects the file download command.

The above described embodiments of the present invention are suitable for conveniently sending confidential electronic mails (e-mails) that are encrypted during the transmission. The addressee of the confidential e-mail may receive a conventional e-mail informing him or her about the availability of the confidential e-mail. The conventional e-mail may advantageously contain the URL of the webpage 22a of the server 20 that is to be visited for receiving the confidential e-mail. The addressee preferably also receives a password for connecting to the webpage 22a, which is preferably communicated to him or her separately (e.g. in an sms, in an e-mail sent to a different e-mail address, orally by the sender of the confidential e-mail, etc.). When the user visits the URL using the web browser 32 of his or her communication device 30 the above described steps are performed with the following particulars. In step 3 the data 37 inputted by the user preferably

includes the password for downloading the confidential e-mail, which password is verified by the server 20 in the course of processing the decrypted client data package 60 in step 6.2. If the password is valid, the confidential e-mail may be sent in the response data package 70 either as a small file in accordance with the steps described in connection with Fig. 2 or as a big file employing symmetric cryptography as explained in connection with Fig. 3. If the addressee wishes to respond the method steps are repeated from step 3: the user inputs new data 37 (in particular a text response), the JavaScript program 34 generates a new asymmetric key pair 36, creates a new client data package 60, which includes the new client public key 36b and the new data 37 inputted by the user, encrypts the new client data package 60 with the server public key 26b (optionally after splitting up the message and/or calculating the new HMAC 61), which is then transmitted to the server 20, where it is decrypted and processed. Any response message to the user's new client data package 60 (e.g. confirmation message, error message, HTML content, etc.) is encrypted with the new client public key 36b and sent back. It is also possible that the JavaScript program 34 uses the symmetric key 74 received and decrypted in Step 9 of the embodiment according to Fig. 2, or that the JavaScript program 34 requests a symmetric key 74 in order to encrypt and send a big file 72.

The method according to the invention is also suitable for sending other encrypted data from the server 20 to the client communication device 30 in other applications such as file transfer, electronic payment information, Websocket, HTTPS proxy or other streaming applications.

WebSockets is a next-generation bidirectional communication technology for web applications which operates over a single socket and is exposed via a JavaScript interface in HTML 5 compliant browsers. Websockets using Socket.send() and Socket.onmessage() to send and receive data. Before Scoket.send() and after Socket.onmessage() the invention is also callable to encrypt or decrypt Websocket data.

The method according to the present invention is also suitable for authentication purposes, in particular for authenticating the client web browser 32. Such an embodiment of the method according to the invention is displayed in Fig. 4 and 5. Since the authentication process is very similar to the above discussed embodiments only the differences will be explained in greater detail here.

According to this embodiment the electronic communication channel 50 established between the server 20 and the client communication device 30 is not necessarily an AJAX connection, instead it can be any HTTP(S) connection.

In step 4.1.1 the JavasScript program 34 preferably stores the generated asymmetric key pair 36 in the local storage 38 of the web browser 32. The server public key 26b may preferably be stored in the local storage 38 of the client web browser 32 as well, whereby the server public key 26b is only accessible by a JavaScript program 34 running on the domain of the HTML website 22a visited in step 2.

The first client data package 60 created in step 4.2 preferably includes identification data 37' of the client web browser 34. The identification data 37' may be a user name, password, e-mail address or any other data or data combination that is suitable for identifying the user, which may be inputted by the user through the HTML form.

After the server 20 receives the encrypted data package 60', decrypts it with the server private key 26a and extracts the client public key 36b and the identification data 37' there from, the latter are stored by the server 20. Following this, the client web browser 32 may connect to the same or to another HTML webpage 22a of the same HTML website 22 (and of the same domain) and may request the content of this HTML webpage 22a in step 7. When the HTML webpage 22a is downloaded in step 8, the JavaScript program 34 is sent to the web browser 34 again. In step 9 the user may input data (such as the identification data 37' containing user name and optionally password) or one or more commands. The JavaScript program 34 is launched in step 10 and performs the following steps. In step 10.1 the JavaScript program 34 reads the client public key 36b from the local storage 38 and uses this and the identification data 37' and possible other data 37 (e.g. data inputted by the user in step 9) to create a second client data package 64 in step 10.2. Optionally the identification data 37' may be sent prior to starting step 10.1, e.g. the user has already logged in with his or her user name (or other identifier), after which the SESSION ID can serve to link the client public key 36b to the user's earlier provided identification data 37'.

Optionally the HMAC 65 of the second client data package 64 may be calculated which is followed by encrypting the second client data package 64 (possibly together with the HMAC 65), and the resulting encrypted data package

64' is sent to the server 22 over the electronic communication channel 50 in step 11.

The encrypted data package 64' is decrypted with the server private key 26a in step 12, and the HMAC 65 is verified in step 12.1 in case such a HMAC has been included in the encrypted data package 64'. The decrypted second client data package 64 is then processed by the server 22 in step 12.2, whereby the client public key 36b and optionally the identification data 37' and other data 37 are extracted. In step 13 the server 22 authenticates the client web browser 32 by comparing the presently received client public key 36b with the earlier received and stored client public key 36b that is associated with the same identification data 37'. If the authentication is successful the server 20 proceeds e.g. by performing a requested operation. If the authentication fails the server 20 redirects the web browser 32 to an error message URL or by aborting in any other way.

The features described in connection with the embodiments of Figs, 2, 2a, 3 are also conceivable in connection with the present embodiment. In particular it is possible to split any of the data packages 60, 64, 70 or to send big files using a symmetric key that is transmitted with asymmetric cryptography.

Various modifications to the above disclosed embodiments will be apparent to a person skilled in the art without departing from the scope of protection determined by the attached claims.

CLAIMS

1. Method for allowing secure communication between a server and a client web browser running on a client communication device which is connectable with the server over an electronic communication channel, **characterised by** comprising:

- providing a HTML webpage at the server
- sending a JavaScript program to the client web browser when the HTML webpage is downloaded by the client web browser, which JavaScript program is configured, when executed by the client web browser, to:
  - generate a client asymmetric key pair consisting of a client public key and a client private key,
  - create a first client data package comprising the client public key, and
  - send the first client data package to the server over the electronic communication channel.

2. The method according to claim 1, **characterised by** comprising the steps of:

- receiving the first client data package at the server,
- extracting the client public key from the first client data package.

3. The method according to claims 1 or 2, **characterised by** that the JavaScript program is further configured, when executed by the client web browser, to include in the first client data package data inputted at the HTML website.

4. The method according to any of the claims 1 to 3, **characterised by** further comprising the steps of:

- providing an asymmetric key pair at the server consisting of a server public key and a server private key,

- making the server public key accessible to the JavaScript program,

and wherein the JavaScript program is further configured, when executed by the client web browser, to encrypt the first client data package with the server public key.

5. The method according to claim 4, **characterised by** further comprising decrypting the first client data package with the server private key at the server.

6. The method according to claims 4 or 5, **characterised by** wherein making the server public key accessible to the JavaScript program comprises sending the server public key to the client web browser when the HTML website is downloaded by the client web browser.

7. The method according to any of the claims 1 to 6, **characterised by** that the JavaScript program is further configured, when executed by the client web browser, to

- split the first client data package into data package portions having sizes that can be encrypted by the server public key,

- encrypt each data package portion with the server public key, and

- send the first client data package as a data array of the encrypted data package portions,

the method further comprising:

- decrypting each encrypted data package portion of the data array at the server, and

- concatenating the split data package portions to obtain the first client data package.

8. The method according to any of the claims 1 to 7, **characterised by** that the JavaScript program is further configured, when executed by the client web browser, to:

- calculate a HMAC of the first client data package and

- concatenate the HMAC with the first client data package,

and the method further comprising:

- extracting the received HMAC at the server,

- calculating a new HMAC of the first client data package, and

- comparing the calculated new HMAC and the extracted received HMAC for tamper proofing the client data package.

9. The method according to any of the claims 1 to 8, **characterised by** that the format of the client asymmetric key pair correspond to the openssl library format.

10. The method according to any of the claims 1 to 9, **characterised by** further comprising, after extracting the client public key from the first client data package,

- creating a server response data package,

- encrypting the server response data package with the client public key, and

- sending the server response data package to the client browser over the electronic communication channel.

11. The method according to claim 10, **characterised by** that the JavaScript program is further configured, when executed by the client web browser, to:

- receive the server response data package,

- decrypt the server response data package with the client private key.

12. The method according to claims 10 or 11, **characterised by** further comprising the steps of:

- splitting the server response data package into data package portions having sizes that can be encrypted by the client public key,

- encrypting each data package portion with the client public key, and

- sending the server response data package as a data array of the encrypted data package portions, and

wherein the JavaScript program is further configured, when executed by the client web browser, to

- decrypt each encrypted data package portion of the data array with the client public key, and

- concatenate the data package portions in order to obtain the server response data package.

13. The method according to claim 12, **characterised by** further comprising the steps of:

- creating a string from the data array by applying JSON coding, and

- sending the JSON coded string to the client browser over the electronic communication channel.

14. The method according to any of the claims 10 to 13, **characterised by** further comprising the steps of:

- generating a symmetric key at the server, and

- including the symmetric key in the server response data package and encrypting the server response data package with the client public key,

- creating a second server data package,

- encrypting the second server data package with the symmetric, and

- sending the second server data package to the client browser over the electronic communication channel,

wherein the JavaScript program is further configured, when executed by the client web browser, to:

- decrypt the server response data package with the client private key and extract the symmetric key from the decrypted server response data package, and

- decrypt the second server data package with the symmetric key.

15. The method according to any of the claims 10 to 14, **characterised by** that the JavaScript program is further configured, when executed by the client web browser, to:

- generate a new asymmetric key pair consisting of a new client public key and a new client private key each time a new client data package is to be sent to the server,

- create the new client data package comprising the new client public key, and

- send the new client data package to the server over the electronic communication channel,

the method further comprising:

- receiving the new client data package at the server,

- extracting the new client public key from the new client data package,

- creating a new server response data package,

- encrypting the new server response data package with the client public key, and

- sending the new server response data package to the client browser over the electronic communication channel.

16. The method according to any of the claims 10 to 15, **characterised by** further comprising:

- opening an AJAX connection between the server and the client web browser over the electronic communication channel, and

- using the AJAX connection to send the client and server data packages.

17. The method according to claim 16, **characterised by** that the JavaScript program is further configured, when executed by the client web browser, to store the generated asymmetric key pair in a local storage of the web browser or in the form of a variable in a memory of the client communication device.

18. The method according to any of the claims 1 to 9, **characterised by** that the JavaScript program is further configured to store the generated asymmetric key pair in a local storage of the client browser

19. The method according to claim 18, **characterised by** further comprising, after extracting the client public key from the first client data package, storing the client public key at the server.

20. The method according to claims 19, **characterised by** further comprising the steps of

- providing a second HTML webpage under the same domain as the first HTML webpage at the server,

- sending a second JavaScript program to the client web browser when the second HTML webpage is downloaded by the client web browser, which JavaScript program, when executed, is configured to

- read the client public key from the local storage,
- create a second client data package comprising the client public key, and
- send the second client data package to the server over the electronic communication channel.

21. The method according to claim 20, **characterised by** further comprising:

- receiving the second client data package at the server,

- extracting the client public key from the second client data package,

- authenticating the client web browser by comparing the received client public key with the client public key stored at the server.

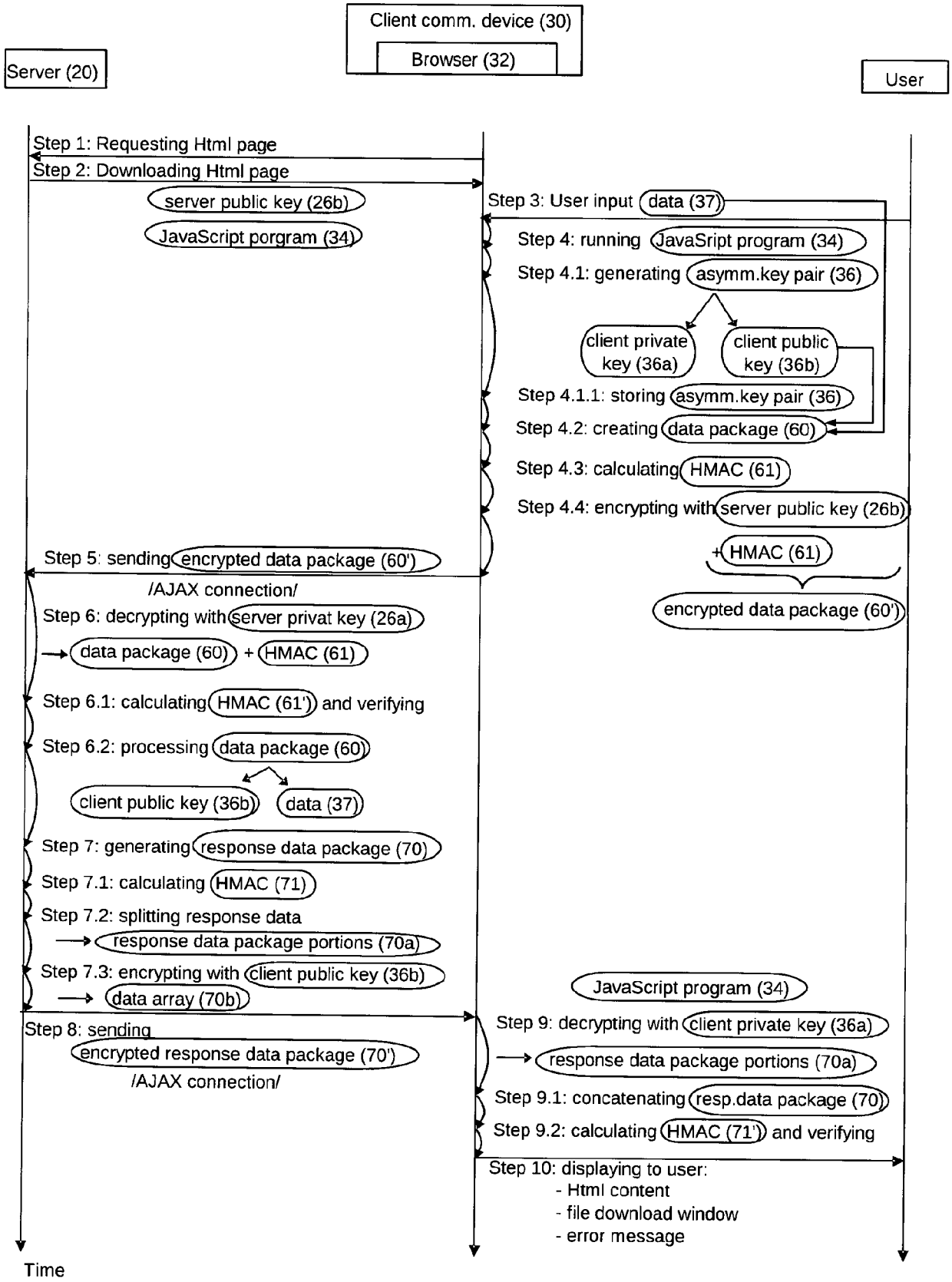

Client comm. device (30)
Browser (32)
Server (20)
User
Step 1: Requesting Html page
Step 2: Downloading Html page
server public key (26b)
JavaScript porgram (34)
Step 3: User input data (37)
Step 4: running JavaSript program (34)
Step 4.1: generating asymm.key pair (36)
client private key (36a)
client public key (36b)
Step 4.1.1: storing asymm.key pair (36)
Step 4.2: creating data package (60)
Step 4.3: calculating HMAC (61)
Step 4.4: encrypting with server public key (26b)
+ HMAC (61)
encrypted data package (60')
Step 5: sending encrypted data package (60')
/AJAX connection/
Step 6: decrypting with server privat key (26a)
data package (60) + HMAC (61)
Step 6.1: calculating HMAC (61') and verifying
Step 6.2: processing data package (60)
client public key (36b)
data (37)
Step 7: generating response data package (70)
Step 7.1: calculating HMAC (71)
Step 7.2: splitting response data
response data package portions (70a)
Step 7.3: encrypting with client public key (36b)
data array (70b)
JavaScript program (34)
Step 8: sending
encrypted response data package (70')
/AJAX connection/
Step 9: decrypting with client private key (36a)
response data package portions (70a)
Step 9.1: concatenating resp.data package (70)
Step 9.2: calculating HMAC (71') and verifying
Step 10: displaying to user:
- Html content
- file download window
- error message
Time

Fig. 2

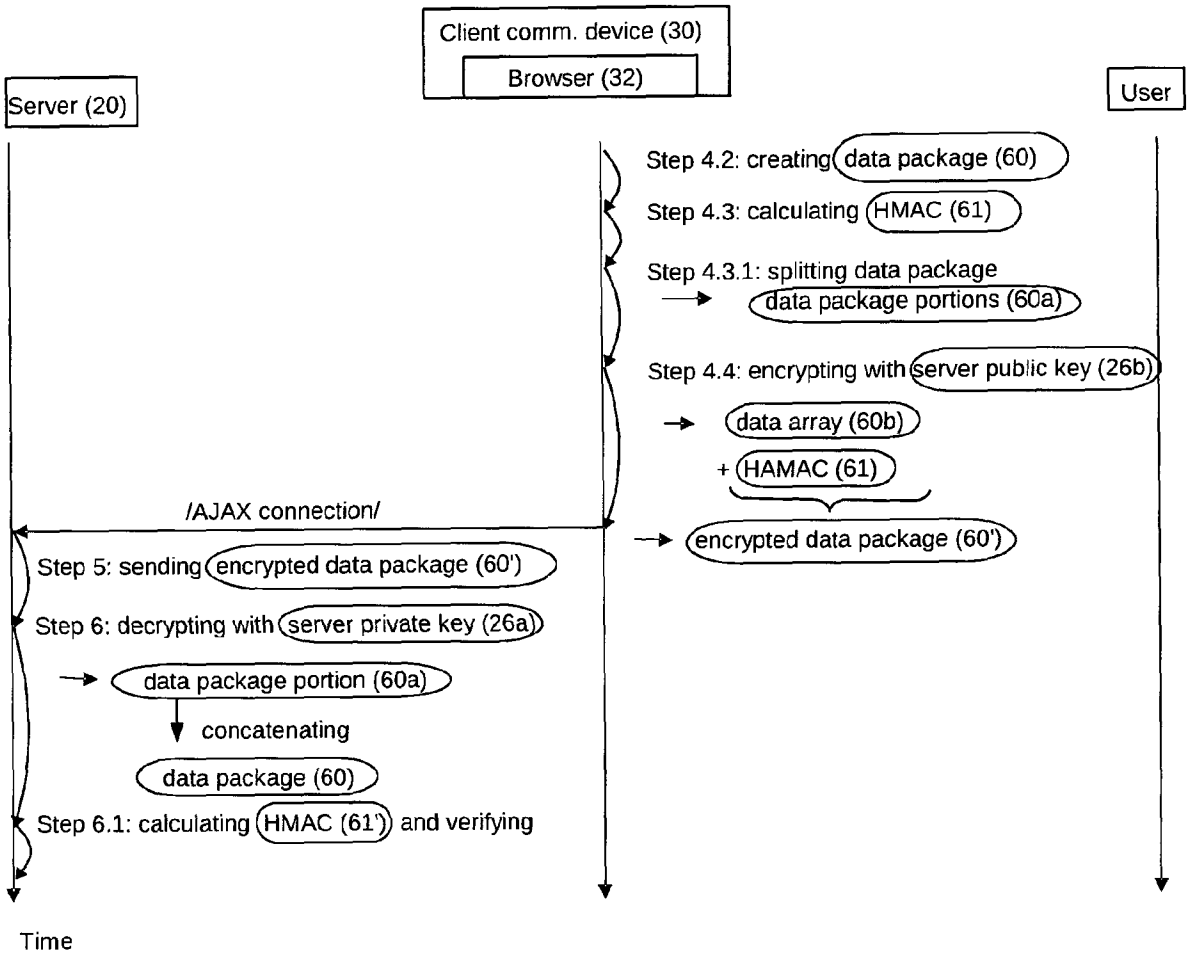

Client comm. device (30)
Browser (32)
Server (20)
User
Step 4.2: creating data package (60)
Step 4.3: calculating HMAC (61)
Step 4.3.1: splitting data package
data package portions (60a)
Step 4.4: encrypting with server public key (26b)
data array (60b)
+ HAMAC (61)
/AJAX connection/
encrypted data package (60')
Step 5: sending encrypted data package (60')
Step 6: decrypting with server private key (26a)
data package portion (60a)
concatenating
data package (60)
Step 6.1: calculating HMAC (61') and verifying
Time

Fig. 2a

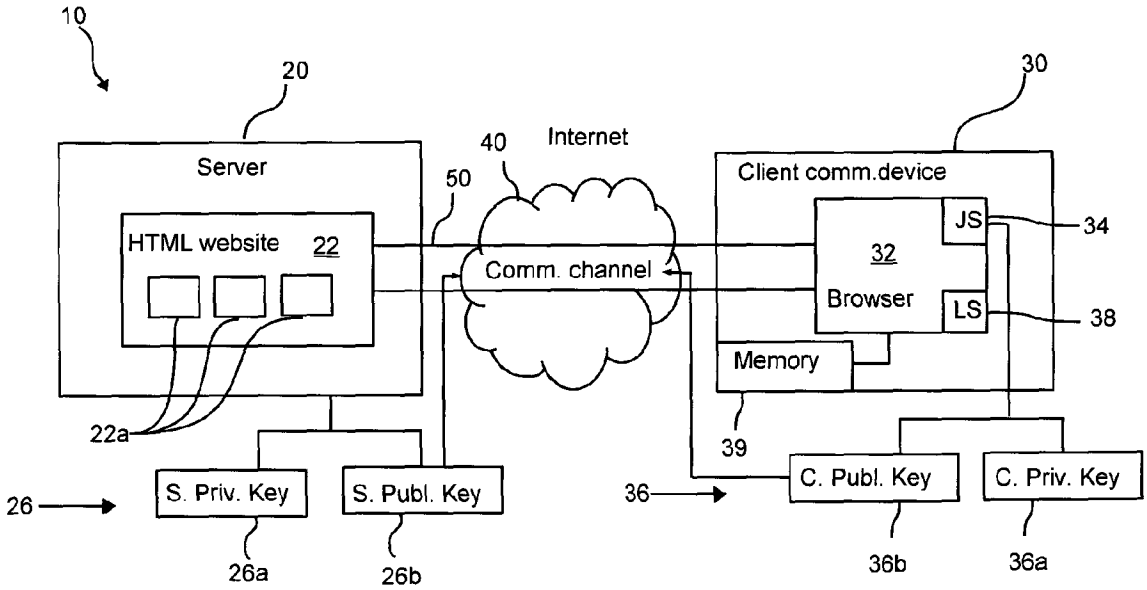

10
20
30
Server
40
Internet
50
Client comm.device
HTML website 22
Comm. channel
JS
34
32
Browser
LS
38
Memory
22a
39
26
S. Priv. Key
S. Publ. Key
36
C. Publ. Key
C. Priv. Key
26a
26b
36b
36a

Fig. 1

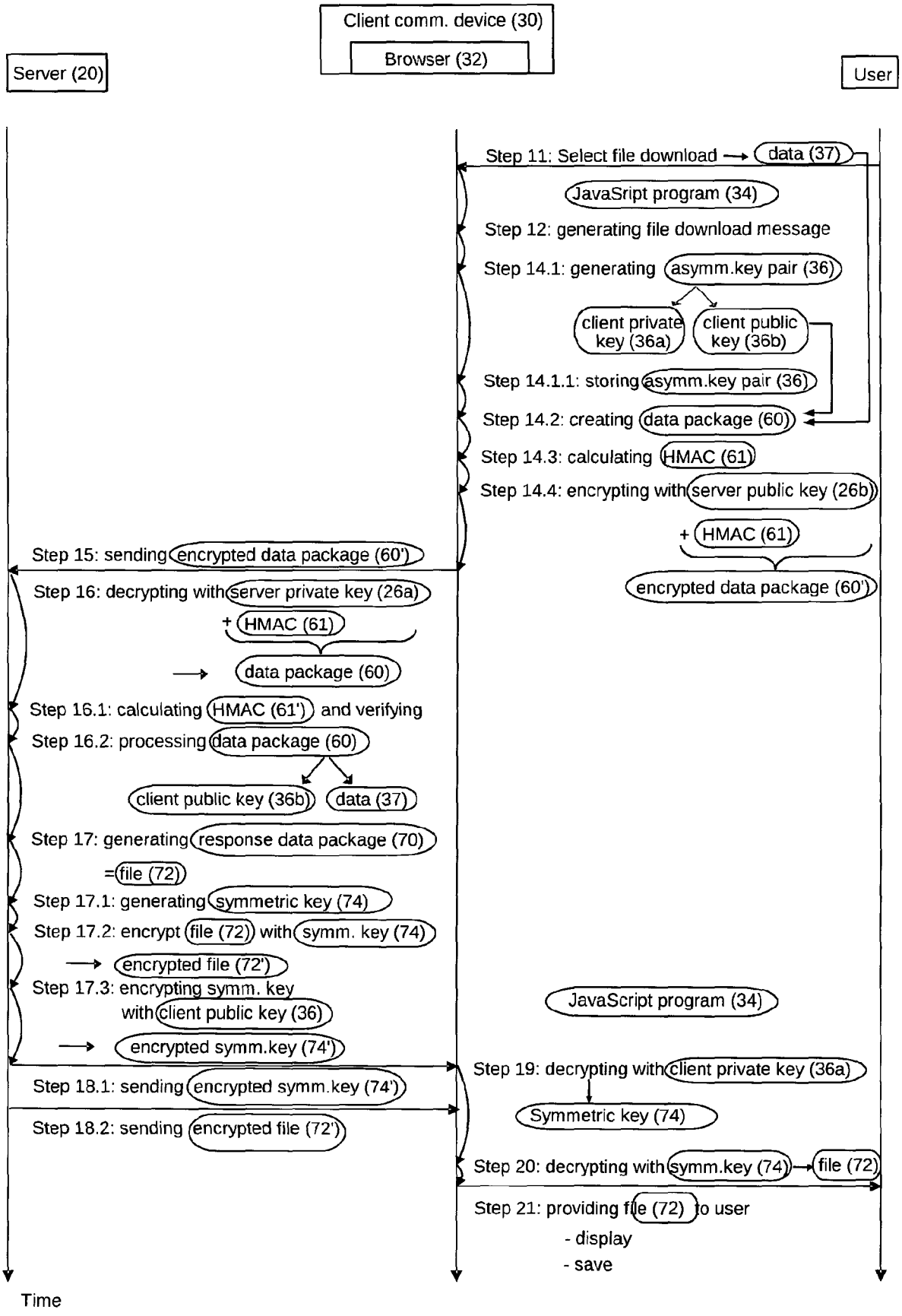

Client comm. device (30)
Browser (32)
Server (20)
User
Step 11: Select file download
data (37)
JavaSript program (34)
Step 12: generating file download message
Step 14.1: generating asymm.key pair (36)
client private key (36a)
client public key (36b)
Step 14.1.1: storing asymm.key pair (36)
Step 14.2: creating data package (60)
Step 14.3: calculating HMAC (61)
Step 14.4: encrypting with server public key (26b)
+ HMAC (61)
encrypted data package (60')
Step 15: sending encrypted data package (60')
Step 16: decrypting with server private key (26a)
+ HMAC (61)
data package (60)
Step 16.1: calculating HMAC (61') and verifying
Step 16.2: processing data package (60)
client public key (36b)
data (37)
Step 17: generating response data package (70)
=file (72)
Step 17.1: generating symmetric key (74)
Step 17.2: encrypt file (72) with symm. key (74)
encrypted file (72')
Step 17.3: encrypting symm. key with client public key (36)
encrypted symm.key (74')
JavaScript program (34)
Step 18.1: sending encrypted symm.key (74')
Step 19: decrypting with client private key (36a)
Symmetric key (74)
Step 18.2: sending encrypted file (72')
Step 20: decrypting with symm.key (74)
file (72)
Step 21: providing file (72) to user
- display
- save
Time

Fig. 3

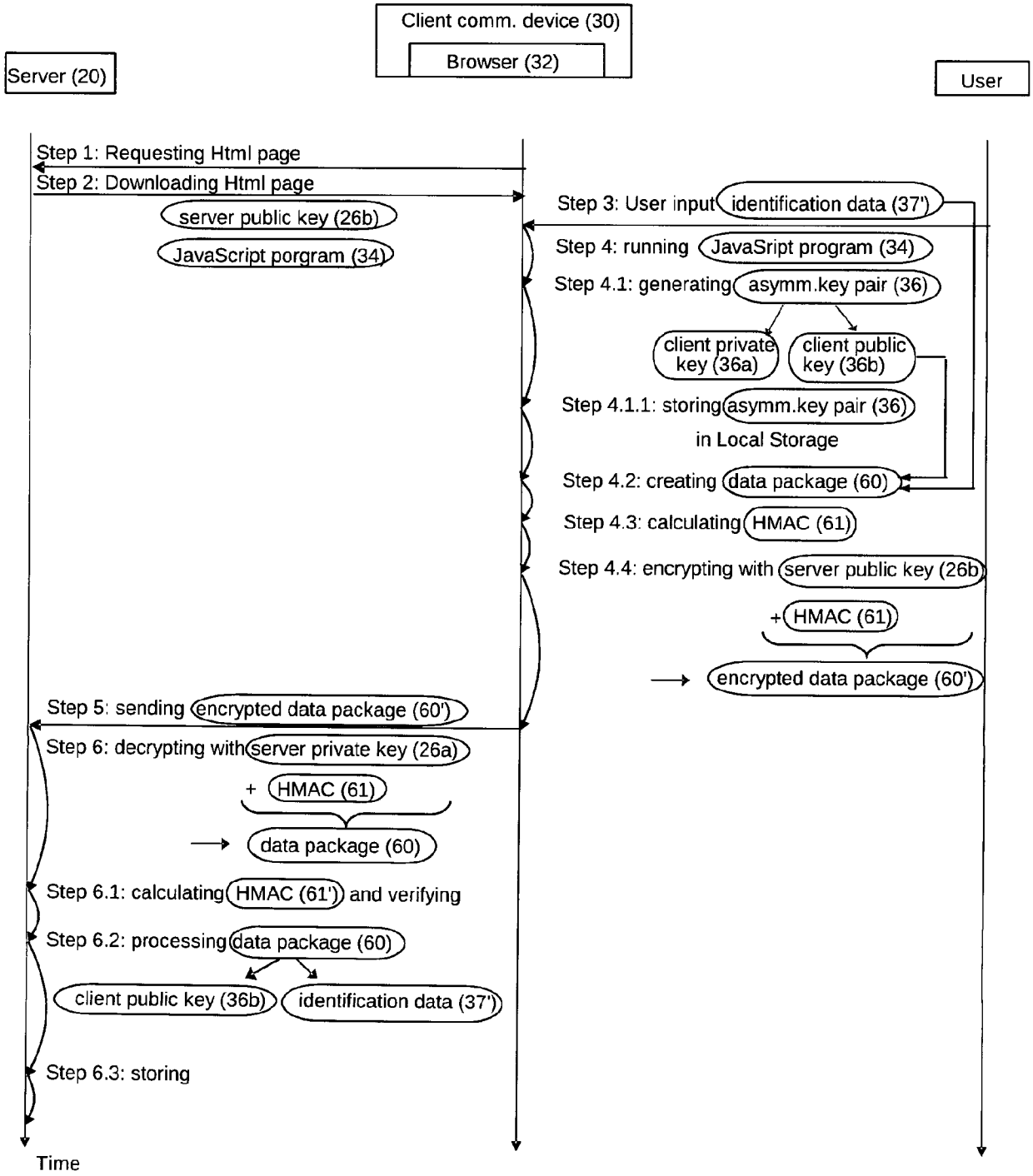

Client comm. device (30)
Browser (32)
Server (20)
User
Step 1: Requesting Html page
Step 2: Downloading Html page
server public key (26b)
JavaScript porgram (34)
Step 3: User input
identification data (37')
Step 4: running
JavaSript program (34)
Step 4.1: generating
asymm.key pair (36)
client private key (36a)
client public key (36b)
Step 4.1.1: storing
asymm.key pair (36)
in Local Storage
Step 4.2: creating
data package (60)
Step 4.3: calculating
HMAC (61)
Step 4.4: encrypting with
server public key (26b)
+
HMAC (61)
encrypted data package (60')
Step 5: sending
encrypted data package (60')
Step 6: decrypting with
server private key (26a)
+
HMAC (61)
data package (60)
Step 6.1: calculating
HMAC (61')
and verifying
Step 6.2: processing
data package (60)
client public key (36b)
identification data (37')
Step 6.3: storing
Time

Fig. 4

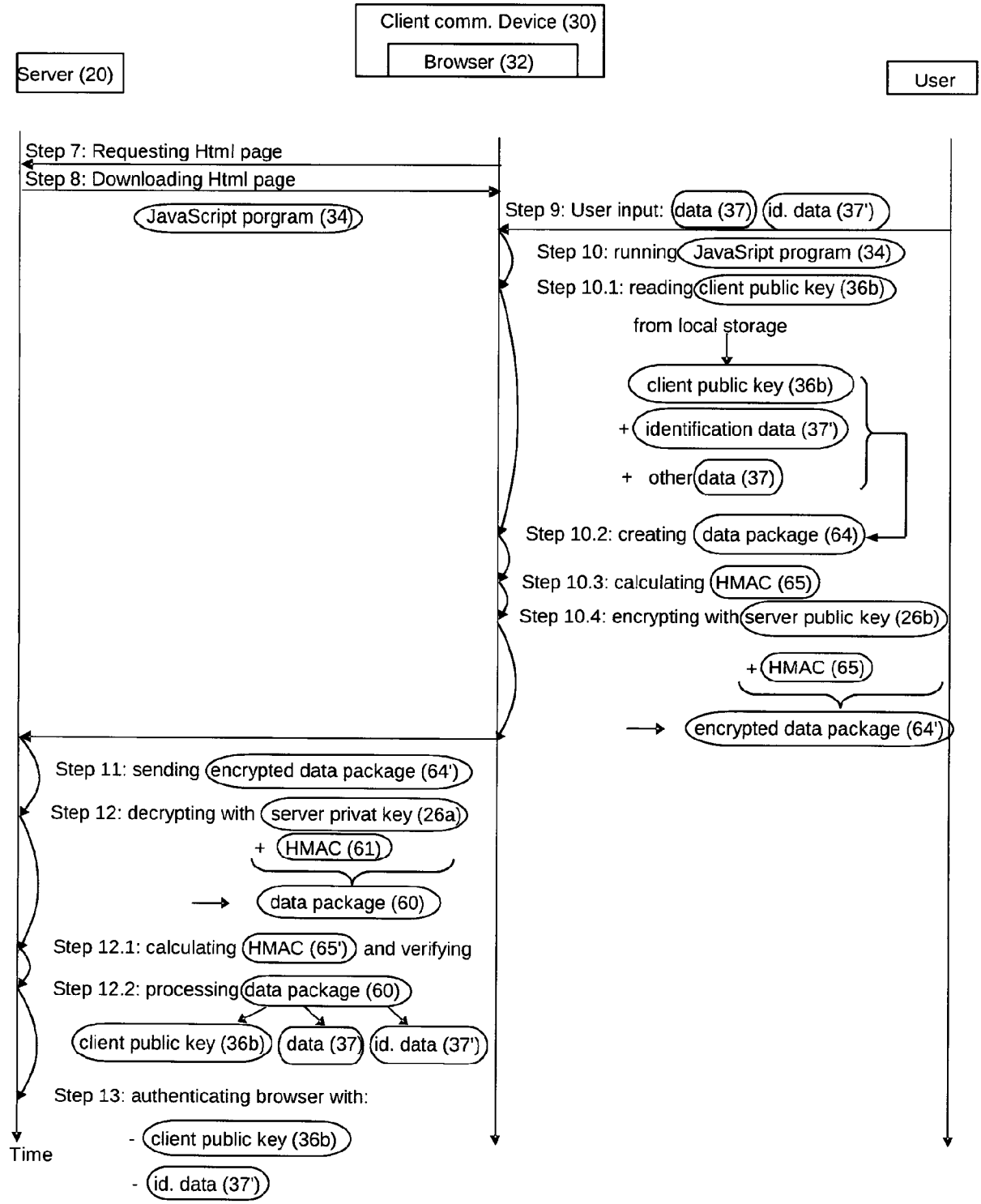

Client comm. Device (30)
Browser (32)
Server (20)
User
Step 7: Requesting Html page
Step 8: Downloading Html page
JavaScript porgram (34)
Step 9: User input: data (37) id. data (37')
Step 10: running JavaSript program (34)
Step 10.1: reading client public key (36b)
from local storage
client public key (36b)
+ identification data (37')
+ other data (37)
Step 10.2: creating data package (64)
Step 10.3: calculating HMAC (65)
Step 10.4: encrypting with server public key (26b)
+ HMAC (65)
encrypted data package (64')
Step 11: sending encrypted data package (64')
Step 12: decrypting with server privat key (26a)
+ HMAC (61)
data package (60)
Step 12.1: calculating HMAC (65') and verifying
Step 12.2: processing data package (60)
client public key (36b)
data (37)
id. data (37')
Step 13: authenticating browser with:
- client public key (36b)
- id. data (37')
Time

Fig. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/HU 2014/000007

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/62 *(2013.01)*
H04L 9/30 *(2006.01)*

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 9/00,9/28-9/32,29/00, 29/12, G06F 3/00, 3/12, 13/00, 13/14, 21/00, 21/60,21/62, H04W 12/00, G06Q 20/00

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 practicable, search terms used)

PatSearch (RUPTO internal), USPTO, PAJ, Esp@cenet, Information Retrieval System of FIPS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	RU 2273107 C2 (ZAKRYTOE AKTSIONERNOE OBSCHESTVO "PLATOFON") 27.03.2006, p. 7, lines 23-25, p. 15, lines 16-30, line 50 - p. 16, line 5, p. 17, lines 4-10, p. 29, lines 28-30, 43-45, p. 40, lines 47-50, p. 45, line 51, p. 46, lines 1-7	1-3
Y	WO 2013/157957 A1 (INVENIA AS) 24.10.2013, p. 9, lines 23-38, p. 10, lines 15-19, p. 13, lines 17-35	1-3

☐ 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
02 October 2014 (02.10.2014)	09 October 2014 (09.10.2014)
Name and mailing address of the ISA/RU: FIPS, Russia, 123995, Moscow, G-59, GSP-5, Berezhkovskaya nab., 30-1 Facsimile No. +7 (499) 243-33-37	Authorized officer I. Kryazhev Telephone No. (499) 240-25-91

Form PCT/ISA/210 (second sheet) (July 2009)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/HU 2014/000007

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☒ Claims Nos.: 4-21
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
☐ No protest accompanied the payment of additional search fees.