



(10) **DE 10 2022 208 220 A1** 2024.02.08

(12)

## Offenlegungsschrift

(21) Aktenzeichen: **10 2022 208 220.6**

(22) Anmeldetag: **08.08.2022**

(43) Offenlegungstag: **08.02.2024**

(51) Int Cl.: **H04L 9/32 (2006.01)**

(71) Anmelder:

**Robert Bosch Gesellschaft mit beschränkter  
Haftung, 70469 Stuttgart, DE**

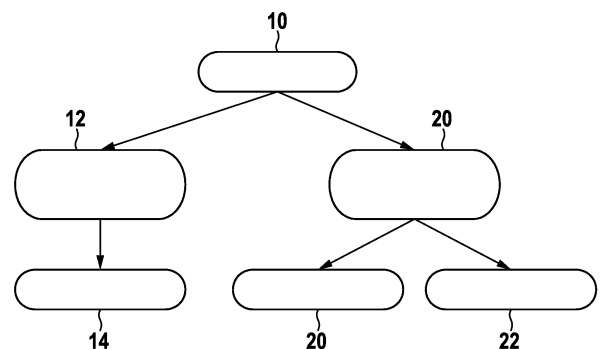
(72) Erfinder:

**Pai, Gautam, 70499 Stuttgart, DE; Pandit, Prajakta,  
70499 Stuttgart, DE**

**Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.**

(54) Bezeichnung: **Authentifizierungsverfahren**

(57) Zusammenfassung: Verfahren zur Authentifizierung eines Clients gegenüber einem Server für einen Diagnosedienst, wobei der Client den Diagnosedienst durch Herstellen einer Verbindung mit einem Server anfordert, indem er ein Zertifikat sendet, das den Client identifiziert, der Server die Anforderung des Clients validiert und auf die Anforderung antwortet, indem er Informationen zum Aufbau eines sicheren Kanals an den Client sendet, der Client die Antwort des Servers validiert und den sicheren Kanal zum Server aufbaut, wenn beim Einleiten der Verbindung mit dem Server durch Senden eines Zertifikats, das den Client identifiziert, eine Präambel zu einem Handshake verwendet wird, um das Problem eines zuvor abgefangenen Client-Zertifikats anzugehen.



**Beschreibung**

**[0001]** Die Erfindung stellt ein Authentifizierungsverfahren eines Clients für einen Diagnosedienst und eine Anordnung für die Durchführung des Verfahrens bereit.

**Bisheriger Stand der Technik**

**[0002]** Unified Diagnostic Services (UDS) ist ein Diagnose-Kommunikationsprotokoll, das für Steuergeräte (Electronic Control Units, ECUs) in der Automobilelektronik verwendet wird. Mithilfe von USD können Diagnosetools alle in einem Fahrzeug bereitgestellten ECUs kontaktieren.

**[0003]** Die UDS-Norm ISO 14229-1:2020 definiert einen neuen Diagnosedienst, den sogenannten Authentifizierungsdienst (29<sub>16</sub>). Die UDS-Norm besteht aus Diagnosediensten, wie etwa das Herunter-/Hochladen von Routinen, das Auslesen bestimmter Speicherorte von einem Server und so weiter, für die eine Authentifizierung erforderlich ist, um die Sicherheit zu gewährleisten. Somit wird der Authentifizierungsdienst (29<sub>16</sub>) als ein Mittel vorgestellt, mit dem der Client seine Identität nachweisen kann, bevor ihm Zugriff auf Daten und/o-der Diagnosedienste gewährt wird, deren Zugriff beschränkt ist.

**Offenlegung der Erfindung**

**[0004]** Gemäß der Erfindung wird ein Authentifizierungsverfahren gemäß Anspruch 1 vorgestellt.

**[0005]** Des Weiteren wird eine zur Durchführung des Verfahrens geeignete Anordnung nach Anspruch 10 vorgestellt.

**[0006]** Das vorgeschlagene Verfahren dient der Authentifizierung eines Clients gegenüber einem Server für einen Diagnosedienst, wobei der Client den Diagnosedienst durch Herstellen einer Verbindung mit einem Server anfordert, indem er ein Zertifikat sendet, das den Client identifiziert, der Server die Anforderung des Clients validiert und auf die Anforderung antwortet, indem er Informationen zum Aufbau eines sicheren Kanals an den Client sendet, der Client die Antwort des Servers validiert und den sicheren Kanal zum Server aufbaut. Beim Einleiten der Verbindung mit dem Server durch Senden eines Zertifikats, das den Client identifiziert, wird eine Präambel zu einem Handshake verwendet, um das Problem eines zuvor abgefangenen Client-Zertifikats anzugehen.

**[0007]** Dementsprechend geht das Verfahren eine Sicherheitslücke an, nämlich das Replay (erneute Wiedergabe) des Client-Zertifikats. In dieser Beschreibung wird insbesondere mit Bezugnahme auf die Zeichnungen diese Sicherheitslücke als Sicherheitslücke Nr. 1 bezeichnet.

**[0008]** In einer Ausführungsform dient das Verfahren dazu, einen Denial-of-Service-Angriff auf den Server zur Erstellung einer Frage abzuschwächen.

**[0009]** In einer anderen Ausführungsform wird das Verfahren dazu verwendet, einen Denial-of-Service-Angriff auf den Server bezüglich eines Eigentumsnachweises abzuschwächen.

**[0010]** Wenn ferner ein nicht authentifiziertes Schlüsselvereinbarungsprotokoll, z. B. das Diffie-Hellman-Schlüsselvereinbarungsprotokoll verwendet wird, umfasst das Verfahren den Schritt der bilateralen Authentifizierung, um das Problem eines nicht authentifizierten kurzlebigen Schlüssels anzugehen, durch den die hergestellte Sitzung authentifiziert wird.

**[0011]** Die Sitzung kann unter Verwendung des Schritts der bilateralen Authentifizierung, die durch eine Option im Authentifizierungsdienst(29<sub>16</sub>) bereitgestellt wird, vor der Ermittlung von Sitzungsschlüsseln hergestellt werden, indem der Client ein zusätzliches Schlüsseltransportprotokoll verwendet, wobei die authentifizierten Schlüssel als die Sitzungsschlüssel oder beispielsweise durch Verwendung eines Schlüsselvereinbarungsprotokolls ermittelt werden, mit dem der Client und Server statische lange Schlüssel oder ein gemeinsames Geheimnis verwenden können, um Sitzungsschlüssel über einen unsicheren Kanal, eine Verschlüsselung mittels Elliptische-Kurven-Diffie-Hellman (ECDH) oder eine Kennwort-authentifizierte Schlüsselvereinbarung zu ermitteln.

**[0012]** In dieser Ausführungsform geht das Verfahren eine Sicherheitslücke an, nämlich die nicht authentifizierten kurzlebigen Schlüssel. In dieser Beschreibung wird insbesondere mit Bezugnahme auf die Zeichnungen diese Sicherheitslücke als Sicherheitslücke Nr. 2 bezeichnet.

**[0013]** In diesem Fall wird das Verfahren zum Abschwächen eines Server-Spoof-Angriffs gegen den Client verwendet.

**[0014]** Die hierin beschriebene Anordnung ist für das Durchführen des vorgeschlagenen Verfahrens geeignet. Die Anordnung kann in Hardware und/oder Software implementiert werden. Darüber hinaus kann die Anordnung in ein elektronisches Steuergerät (ECU) integriert werden oder kann in einer Ausführungsform selber ein ECU sein.

#### Kurzbeschreibung der Zeichnungen

**Fig. 1** zeigt eine Übersicht über den Authentifizierungsdienst.

**Fig. 2** zeigt ein Flussdiagramm, das einen durch die UDS-Norm eingeführten Authentifizierungsdienst (29<sub>16</sub>) veranschaulicht (Dienst 29).

**Fig. 3** zeigt das Flussdiagramm von **Fig. 1**, das eine erste Sicherheitslücke Nr. 1 in dem Dienst 29 veranschaulicht.

**Fig. 4** zeigt das Flussdiagramm von **Fig. 1**, das eine zweite Sicherheitslücke Nr. 2 in dem Dienst 29 veranschaulicht.

**Fig. 5** zeigt ein überarbeitetes Flussdiagramm auf der Grundlage eines Flussdiagramms von **Fig. 1**, das eine Abschwächung der Sicherheitslücke im Rahmen von DoS veranschaulicht.

**[0015]** Es versteht sich, dass die vorstehend genannten und die nachstehend noch zu erläuternden Merkmale nicht nur in der jeweils angegebenen Kombination, sondern auch in anderen Kombinationen oder in Alleinstellung verwendbar sind, ohne vom Umfang der Erfindung abzuweichen.

**[0016]** Die Erfindung ist anhand von Ausführungsbeispielen in den Zeichnungen schematisch dargestellt und wird nachfolgend unter Bezugnahme auf die Zeichnungen näher erläutert. Es versteht sich, dass die Beschreibung in keiner Weise den Umfang der vorliegenden Erfindung einschränkt und fast eine Veranschaulichung von Ausführungsformen der Erfindung darstellt.

#### Beschreibung von Ausführungsformen

**[0017]** **Fig. 1** zeigt eine Übersicht über den Authentifizierungsdienst. Der Authentifizierungsdienst (29<sub>16</sub>) 10 unterstützt zwei Sicherheitskonzepte:

Ein erstes Konzept Nr. 112 basiert auf Prozeduren über die Public-Key-Infrastruktur (PKI) mittels Zertifikatsaustausch (PKI Certificate Exchange, APCE) unter Verwendung von asymmetrischer Kryptografie 14.

**[0018]** Ein zweites Konzept Nr. 220 basiert auf einem Frage-Antwort-Verfahren (Challenge Response, ACR) ohne PKI-Zertifikate, das entweder asymmetrische Kryptografie 22 oder symmetrische Kryptografie 24 verwendet.

**[0019]** Für das hier beschriebene Verfahren wird das erste Sicherheitskonzept Nr. 112 betrachtet. Zur Umsetzung dieses Konzepts wird ein wie das durch die UDS-Norm vorgestellte detaillierte Flussdiagramm für den Authentifizierungsdienst (29<sub>16</sub>) herangezogen, wie in **Fig. 2** gezeigt.

**[0020]** **Fig. 2** zeigt auf der linken Seite die von einem Client 100 durchgeführten Schritte und auf der rechten Seite die von einem Server 102 durchgeführten Schritte. Zwischen diesen beiden Parteien 100 und 102 zeigt **Fig. 2** die Kommunikationen, die zwischen dem Client 100 und dem Server 102 ausgetauscht werden.

**[0021]** Die UDS-Norm in dem Authentifizierungsdienst (29<sub>16</sub>) stellt mehrere Optionen innerhalb des ersten Konzepts Nr. 1 bereit. Eine solche Option ist die Auswahl zwischen unidirektionaler und bidirektionaler Authentifizierung. Bei der unidirektionalen Authentifizierung muss sich nur der Client 100 gegenüber dem Server 102 authentifizieren. Bei der bidirektionalen Authentifizierung müssen sich sowohl der Client 100 als auch der Server 102 gegenseitig authentifizieren. Bei Konzept Nr. 1 verwendet der Mechanismus zum Verifi-

zieren der Authentizität der kommunizierenden Einheit typischerweise PKI-Zertifikate. Der Dienst unterstützt auch verschiedene Mechanismen zum Aufbau eines sicheren Kanals, um die Vertraulichkeit und Integrität der Nachrichten zu gewährleisten.

**[0022]** Der Authentifizierungsprozess startet bei Schritt 110. Der Client 100 leitet eine Verbindung mit dem Server 102 ein und stellt die notwendigen Sicherheitsinformationen bereit, wie etwa die Zertifikate und Optionen zum Konfigurieren des sicheren Kanals, falls erforderlich, und so weiter (Schritte 112 und 114).

Schritt 112 Frage-Client erstellen

Schritt 114 Zertifikat-Client, Frage-Client senden (Authentifizierungsdienst mit Unterfunktion verifyCertificateUnidirectional oder verifyCertificate Bidirection)

**[0023]** Er kann auch die Serverauthentifizierung anfordern. Der Server 102 validiert diese Client-Anforderung (Schritt 116) und fährt mit den erforderlichen Schritten fort (Schritte 118, 120, 122, 124, 126, 128, 130).

Schritt 118 Frage-Server erstellen

Schritt 120 Server für kurzlebige privates/öffentliches Schlüsselpaar erzeugen

Schritt 122 Eigentumsnachweis-Server berechnen = signieren (Server für kurzlebigen öffentlichen Schlüssel, Frage-Server, Frage-Client)

Schritt 124 Frage-Server, Zertifikat-Server, Eigentumsnachweis-Server senden

Schritt 126 Eigentumsnachweis-Client verifizieren

Schritt 128 Sitzungsschlüssel erstellen oder ableiten und Zugriffsrechte für Sitzungsschlüsselinformationen-Satz einrichten

Schritt 130 Antwort und Sitzungsschlüsselinformationen senden

**[0024]** Der Server 102 kann Sicherheit bereitstellen, die für den Aufbau eines sicheren Kanals mit dem Client 100 wesentlich ist. Der Client 100 validiert die Antwort (Schritt 132) und fährt mit den Schritten fort, um denselben sicheren Kanal mit dem Server 102 aufzubauen (Schritte 134, 136, 138, 140, 142, 144).

Schritt 134 Client für kurzlebige privates/öffentliches Schlüsselpaar erzeugen

Schritt 136 Eigentumsnachweis-Server berechnen = signieren (Server für kurzlebigen öffentlichen Schlüssel, Frage-Client, Frage-Server)

Schritt 138 Client für Eigentumsnachweis, Client für kurzlebigen öffentlichen Schlüssel senden (Authentifizierungsdienst mit Unterfunktion proofofownership)

Schritt 140 Sitzungsschlüssel extrahieren oder ableiten

Schritt 142 Sitzungsschlüsselinformationen verifizieren

Schritt 144 Sitzungsschlüssel aktivieren

**[0025]** Das Flussdiagramm umfasst ferner die Kommunikationen, die zwischen dem Client 100 und dem Server 102 ausgetauscht werden.

```
Anforderungsnachricht für die Kommunikation 170
  < Kommunikationskonfiguration >
  < Zertifikats-Client>
  < Frage-Client>
Antwortnachricht für die Kommunikation 172
  < Rückgabewert>
Antwortnachricht für die Kommunikation 174
  < Rückgabewert>
  < Frage-Server>
  < Zertifikats-Server>
  < Eigentumsnachweis-Server>
  <Server für kurzlebigen öffentlichen Schlüssel>
```

Anforderungsnachricht für die Kommunikation 176  
 < Eigentumsnachweis-Client>  
 <Client für kurzlebigen öffentlichen Schlüssel>  
 Antwortnachricht für die Kommunikation 178  
 < Rückgabewert>  
 Antwortnachricht für die Kommunikation 180  
 < Rückgabewert>  
 <Sitzungsschlüsselinformationen>

**[0026]** Des Weiteren zeigt das Flussdiagramm die Schritte, die von dem Client 100 und dem Server 102 im Rahmen des hierin beschriebenen Flusses durchgeführt werden.

Schritt 133                      Zertifikats-Server und Eigentumsnachweis OK?

Andernfalls ist die Authentifizierung fehlgeschlagen 150. In diesem Fall mit Schritt 134 fortfahren.

Schritt 143                      Sitzungsschlüsselinformationen OK?

Andernfalls ist die Authentifizierung fehlgeschlagen 150. In diesem Fall fährt das Verfahren mit dem Schritt 144 fort, und dann war die Authentifizierung erfolgreich 152.

Schritt 117                      Zertifikats-Server gültig?

Andernfalls wird die Kommunikation 172 gesendet. In diesem Fall mit Schritt 118 fortfahren.

Schritt 127                      Client für Eigentumsnachweis OK?

**[0027]** Andernfalls wird die Kommunikation 178 gesendet. In diesem Fall mit Schritt 128 fortfahren.

**[0028]** Bei der Untersuchung des in **Abb. 2** gezeigten Verfahrens wurden Sicherheitslücken in diesem Authentifizierungsdienst identifiziert und die Sicherheitsangriffe aufgelistet, die als Auswirkung dieser Sicherheitslücken auftreten können. Bei den Sicherheitslücken und -angriffen handelt es sich um solche, wie in den **Fig. 3** und **4** gezeigt.

**[0029]** **Fig. 3** veranschaulicht die Sicherheitslücke Nr. 1 in Dienst 29.

**[0030]** Es gibt die nachfolgend aufgeführten Sicherheitslücken:

i. Sicherheitslücke Nr. 1: Replay des „Zertifikats-Clients“ (Bezugszeichen 200)

In dieser Figur beinhaltet der Schritt 116 nur die Verifizierung der Vertrauenskette. Der Client (Tester) 100 wird im Rahmen dieses Schrittes nicht authentifiziert. Ein Angreifer kann diese Schwäche ausnutzen und jedes beliebige abgefangene Client-Zertifikat erneut wiedergeben. Da der Server (ECU) 102 zu diesem Zeitpunkt über keine anderen Informationen verfügt, um die Identität des Testers in Schritt 116 zu verifizieren, würde dies dem Angreifer einen unbefugten Zugriff auf die ECU-Ressourcen verschaffen, um die folgenden Angriffe durchzuführen:

a. Sicherheitsangriff Nr. 1: Denial-of-Service (DoS) auf Server 102 für „Frage-Server“ (Bezugszeichen 202)

Nachdem der Angreifer eine nicht autorisierte authentifizierte Sitzung mit dem ECU erlangt hat, kann er das ECU auffordern, eine „Frage“ zu berechnen. Dies ist in Schritt 118 von **Fig. 3** angegeben. Wenn die Anforderungen an die Frage-Berechnung streng sind, um Sicherheit zu gewährleisten, wie etwa, dass die Frage kryptografisch berechnet werden muss, die Frage ein Kriterium für die Mindestlänge erfüllen muss, die Frage zufällig sein muss und die Erzeugung nicht vorhersagbar sein sollte und so weiter, führt dies zu einer Erschöpfung der ECU-Ressourcen. Somit kann ein Angreifer mit einer abgefangenen Client-Zertifikatsinformation DoS-Angriffe auf das ECU durchführen. Da das Client-Zertifikat nicht als vertraulich behandelt wird, kann ein Angreifer diese Information leicht erhalten, was den Replay-Angriff noch stärker begünstigt.

b. Sicherheitsangriff Nr. 2: DoS auf Server für „Eigentumsnachweis-Server“ (Bezugszeichen 204)

Wie bezüglich des vorstehenden Sicherheitsangriffs erläutert, kann ein Angreifer, sobald er eine nicht

autorisierte authentifizierte Sitzung mit dem ECU erfolgreich erlangt hat, DoS-Angriffe problemlos durchführen. Eine weitere Angriffsfläche, die durch den Protokollaufbau bereitgestellt wird, findet sich in Schritt 122. Dieser Schritt 122 beinhaltet asymmetrische kryptografische Berechnungen. Die Berechnungszyklen und der Berechnungsaufwand sind für eine asymmetrische Kryptografie (digitale Signaturen) aufgrund der Schlüssellänge des verwendeten Algorithmus sehr hoch. Dies wird ebenfalls vor einer Authentifizierung eines jeglichen Testers durchgeführt. Somit ergibt sich für einen Angreifer eine weitere Gelegenheit, die ECU-Ressourcen zu missbrauchen und einen DoS-Angriff durchzuführen.

**[0031]** Ein Denial-of-Service-Angriff (DoS) ist ein Cyberangriff, bei dem der Täter versucht, einen Rechner oder eine Netzwerkressource für die vorgesehenen Benutzer nicht verfügbar zu machen, indem er die Funktion eines mit dem Internet verbundenen Hosts vorübergehend oder auf unbestimmte Zeit unterbricht. In der Regel wird dies erreicht, indem der Rechner oder das Netzwerk mit unnötigen Anforderungen überflutet wird, um den Rechner oder das Netzwerk zu überlasten und zu verhindern, dass einige oder alle normalen Anfragen beantwortet werden.

**[0032]** Fig. 4 veranschaulicht die Sicherheitslücke Nr. 2 in Dienst 29.

ii. Sicherheitslücke Nr. 2: nicht authentifizierte kurzlebige Schlüssel (Bezugszeichen 300)

Die UDS-Norm bietet die Möglichkeit, zwischen unidirektionaler und bidirektionaler Authentifizierung zu wählen. Der Hauptunterschied besteht darin, dass sich bei der bidirektionalen Authentifizierung sowohl der Client 100 als auch der Server 102 gegenseitig authentifizieren müssen, während sich bei der unidirektionalen Authentifizierung nur der Client 100 gegenüber dem Server 102 authentifizieren muss. Wenn dann eine sichere Diagnosekommunikation gewünscht wird, bietet die UDS-Norm auch die Option, ein Schlüsselvereinbarungsprotokoll zu verwenden, um die Sitzungsschlüssel abzuleiten.

Diese Schwachstelle wird in den UDS-Dienst eingeführt, wenn eine unidirektionale Authentifizierung zusammen mit Ephemeral Diffie-Hellman als Schlüsselvereinbarungsprotokoll gewählt wird. Das Diffie-Hellman-Schlüsselvereinbarungsprotokoll ist ein Verfahren zum sicheren Austausch kryptografischer Schlüssel über einen öffentlichen Kanal. Die Diffie-Hellman-Schlüsselvereinbarung selbst ist ein nicht authentifiziertes Schlüsselvereinbarungsprotokoll. Es gestattet einem Angreifer, den folgenden Angriff durchzuführen:

a. Sicherheitsangriff Nr. 3: Server-Spoof-Angriff gegen Client

Die Diffie-Hellman-Schlüsselvereinbarung beruht auf der Verschlüsselung mit einem öffentlichen Schlüssel, um die Authentizität sicherzustellen. Im Fall einer unidirektionalen Authentifizierung wird nur der Client (Tester) 100 gegenüber dem Server (ECU) 102 authentifiziert, und somit sind die von dem ECU (in Schritt 120) erzeugten Schlüsselinformationen weiterhin nicht authentifziert. Dies erlaubt es einem Angreifer, sich als ECU auszugeben und böswillige Antworten an den Tester zurückzusenden. Solche Angriffe werden in einem Szenario noch kritischer, in dem der Tester nicht in der Lage ist, das ECU, mit dem er verbunden ist, physisch zu authentifizieren oder zu verifizieren, beispielsweise mit Funktionen wie Firmware-over-the-air (FOTA)-Updates, Ferndiagnosen etc.

**[0033]** Bei dem Verfahren nach Fig. 4 sind die Schritte 112, 122, 132 und 133 optionale Schritte.

**[0034]** Es gilt zu berücksichtigen, dass das Automobilunternehmen zunehmend in das Internet der Dinge (Internet of Things, IoT) integriert wird. Dadurch ist ein Automobil kein isoliertes einzelnes System mehr, sondern wird Teil der vernetzten Welt. Dies hat dazu geführt, dass verschiedene Funktionen eingeführt wurden, die den Zugriff auf verschiedene drahtlose Technologien erfordern. Einige dieser in der Automobilindustrie weit verbreiteten Funktionen sind Software-Updates über eine Funkschnittstelle (Over-the-Air), Ferndiagnose, Connected Services etc. Somit wird es für die Automobilindustrie entscheidend sein, die Sicherheitskonzepte und Schutzmechanismen zu verbessern.

**[0035]** Die ISO-Gemeinschaft (Internationale Organisation für Normung) hat aus ähnlichen Gründen Sicherheitserweiterungen für die bestehenden UDS-Norm eingeführt. Dies verfolgt den wesentlichen Zweck, die Einschränkungen der in der bestehenden Norm bereitgestellten Sicherheitsmechanismen zu überwinden und die Sicherheit zu verbessern. Mit der Zunahme von Anwendungen wie Ferndiagnose und OTA (Over-the-Air) erhöht sich die Angriffsfläche und dadurch auch das Sicherheitsrisiko. Somit muss sichergestellt werden, dass der als Sicherheitsmechanismus eingeführte Dienst keine weiteren Sicherheitsrisiken einführt. Das vorgeschlagene Verfahren zielt darauf ab, einige dieser Sicherheitslücken zu identifizieren und abzuschwächen.

**[0036]** Das vorgeschlagene Verfahren zielt darauf ab, die nachstehenden Sicherheitslücken zu beheben, die aufgrund von Designfehlern im Authentifizierungsdienst in der neuesten UDS-Norm eingeführt wurden.

i. Sicherheitslücke Nr. 1: Replay des „Zertifikats-Client“ (Bezugszeichen 200 in **Fig. 3**)

**[0037]** Um diese Schwachstelle zu beseitigen, wird nachfolgend eine Präambel zum bestehenden Handshake vorgeschlagen. Die folgenden Angriffe können dadurch abgeschwächt werden.

a. Sicherheitsangriff Nr. 1: DoS auf Server für „Frage-Server“ (Bezugszeichen 202)

b. Sicherheitsangriff Nr. 2: DoS auf Server für „Eigentumsnachweis-Server“ (Bezugszeichen 204)

**[0038]** Auch dieser Vorschlag zielt darauf ab, die Änderungen des grundlegenden Handshakes im Vergleich zur UDS-Norm möglichst gering zu halten.

ii. Sicherheitslücke Nr. 2: Nicht authentifizierte kurzlebige Schlüssel (Bezugszeichen 300 in **Fig. 4**)

Um diese Schwachstelle zu beseitigen, wird nachfolgend ein detaillierter Vorschlag zur Beschränkung des Sicherheitsdesigns gemacht. Mit dieser zusätzlichen Einschränkung werden die folgenden Angriffe abgeschwächt.

a. Sicherheitsangriff Nr. 3: Server-Spoof-Angriff gegen Client

**[0039]** Der Aufbau und die Funktion des vorgeschlagenen Verfahrens werden nachstehend zusammen mit alternativen Ausführungsformen beschrieben.

**[0040]** Die Abschwächung der vorstehend genannten Sicherheitslücken wird nachfolgend beschrieben:

i. Sicherheitsmaßnahme für die Sicherheitslücke Nr. 1 (Bezugszeichen 200)

Die Hauptursache für die Einführung dieser Schwachstelle ist die Tatsache, dass der Protokollaufbau einen hohen Rechenaufwand auf der Seite des Servers (ECU) vor der Authentifizierung des Clients (Tester) erfordert. Daher ist das ECU nicht in der Lage, zwischen rechtmäßigen (authentifizierten) und unrechtmäßigen Anforderungen zu unterscheiden.

**[0041]** In diesem Vorschlag wird ein kleiner Handshake zwischen dem Client (Tester) und dem Server (ECU) eingeführt, bevor die ECU-Ressourcen für umfangreiche kryptografische Berechnungen verbraucht werden. Dieser Handshake würde das ECU dabei unterstützen, zwischen einem rechtmäßigen Tester und einem Angreifer zu unterscheiden. Der Vorschlag zielt darauf ab, eine einzige Lösung für beide oben beschriebenen Sicherheitsangriffe einzuführen, ohne drastische Anpassungen der Norm zu erfordern. Dieser Vorschlag kann auch als Vorbedingungsprüfung vor dem Schritt 112 in **Fig. 3** betrachtet werden. Dieser Handshake-Vorschlag beinhaltet den Austausch von Informationen, die in Schritt 114 das Replay der sensiblen Informationen (Zertifikats-Client) an das ECU verhindern würden.

**[0042]** Der Vorschlag ist in **Fig. 5** ausführlich beschrieben.

**[0043]** Bei den weiteren Schritten und Kommunikationen, die in **Fig. 5** im Vergleich zu **Fig. 2** bis **5** dargestellt sind, handelt es sich um:

|             |                                            |
|-------------|--------------------------------------------|
| Schritt 400 | Hallo-Anforderung                          |
| Schritt 404 | Cookie = Hash (ECUsecret, TimestampServer) |
| Schritt 406 | TimestampServer, Cookie senden             |
| Schritt 408 | Cookie gültig?                             |

Falls nein, wird die Kommunikation 424 gesendet. Andernfalls wird das Verfahren bei Schritt 116 fortgesetzt.

|             |                                                                           |
|-------------|---------------------------------------------------------------------------|
| Schritt 410 | weitere Verarbeitung gemäß dem Authentifizierungsdienstfluss der UDS-Norm |
|-------------|---------------------------------------------------------------------------|

Kommunikation 420 Antwortnachricht  
 <TimestampServer>  
 <Cookie>

Kommunikation 422 Anforderungsnachricht

```
< Kommunikationskonfiguration>
< Zertifikats-Client>
< Frage-Client>
<TimestampServer>
<Cookie>
```

Kommunikation 424 Antwortnachricht

```
< Rückgabewert>
```

Kommunikation 426 Antwortnachricht

```
< Rückgabewert>
```

Kommunikation 428 Antwortnachricht

```
< Rückgabewert>
< Frage-Server>
< Zertifikats-Server>
< Eigentumsnachweis-Server>
<Server für kurzlebigen öffentlichen Schlüssel>
```

**[0044]** In einer Ausführungsform umfasst das Verfahren den Schritt der Berechnung eines „Cookies“ seitens des ECU, bevor das ECU selbst bei den anspruchsvollen kryptografischen Berechnungen teilnimmt. Ein Hash-Algorithmus ist ein möglicher Algorithmus, da dieser im Vergleich zu anderen kryptografischen Algorithmen keine großen ECU-Ressourcen verbrauchen würde. Das „Cookie“ wird vorwiegend als Hash mit zwei Eingaben berechnet:

- a. ECU-Geheimnis: Dieser Wert kann über die Hardwareeigenschaften des ECU abgeleitet werden und ist nur dem ECU bekannt. Dieses Geheimnis sollte weder ausgelesen noch geteilt werden. Ohne die Information des „ECU-Geheimnisses“ wäre es rechnerisch nicht möglich, das Cookie zu erzeugen. Da das „ECU-Geheimnis“ zudem unter Verwendung der ECU-Eigenschaften abgeleitet wird, ist das erzeugte Cookie außerdem spezifisch für das ECU, und das Replay der Cookie-Informationen von einem kompromittierten ECU an andere ECUs ist dann nicht mehr zweckmäßig.
- b. Zeitstempel: Diese Information ist enthalten, um Angreifer daran zu hindern, zuvor abgefangene Cookies an ein Ziel-ECU zu senden. Für die Gültigkeit des Zeitstempels kann ein Zeitfenster definiert werden.

**[0045]** Die primäre Absicht bei der Auswahl dieser Eingabewerte für die „Cookie“-Berechnung besteht darin, keinen Zustand für das ECU vorzugeben. Falls also ein Angreifer versucht, DoS durchzuführen, wäre das ECU nicht an der kryptografischen Berechnung beteiligt. Gemäß der UDS-Norm würde das ECU mit der Verifizierung des Client-Zertifikats (Schritt 116) erst dann beginnen, wenn der Tester in Schritt 114 von **Fig. 5** ein gültiges Cookie bereitstellt.

**[0046]** Gemäß dem Vorschlag in **Fig. 5** zeigt ein Tester dem ECU an, dass er den Authentifizierungsdienst einleiten möchte (Schritt 400). Das ECU antwortet dem Tester mit einem „Cookie“ (Schritt 400). Von dem Tester wird dann erwartet, dieses „Cookie“ in Schritt 114 zusammen mit den Zertifikatsinformationen anzuhängen. Die Aufnahme des Cookies in diesen Schritt sichert dem ECU zu, dass es sich um eine legitime Anfrage handelt. Somit wird das ECU durch die Einführung des Cookie-Konzepts als Vorbedingung im Authentifizierungsdienst davon befreit, die anspruchsvollen kryptografischen Berechnungen in Schritt 118 und Schritt 122 durchzuführen. Somit werden die beiden Angriffe, also der Sicherheitsangriff Nr. 1 202 und der Sicherheitsangriff Nr. 2 204, abgeschwächt.

**[0047]** Zusätzlich kann das vorgeschlagene Verfahren Sicherheit für die neu eingeführten Parameter im Handshake bieten. Diese werden nachfolgend aufgeführt:

- a. Vermeidung eines Replays des Cookies: Ein Angreifer, dem es gelungen ist, die in Schritt 114 von **Fig. 5** ausgetauschten Informationen (Cookie + Client-Zertifikat) abzufangen, kann keinen Replay-Angriff mehr durchführen und keine nicht autorisierte authentifizierte Sitzung mit dem ECU starten, da das ECU das Cookie aufgrund dessen abgelaufenen Zeitstempels ungültig machen würde.
- b. Vermeidung der Manipulation des Zeitstempels: Ein Angreifer ist dann auch nicht in der Lage, den Zeitstempel zu manipulieren und das Cookie wiederzugeben. Dies liegt daran, dass die Cookie-Berechnung durch das ECU aufgrund des falschen Zeitstempelwerts fehlschlagen würde.

## ii. Sicherheitsmaßnahme für die Sicherheitslücke Nr. 2 (Bezugszeichen 300)

**[0048]** Wie vorstehend beschrieben führte die UDS-Norm einen neuen Authentifizierungsdienst ein, um die Sicherheit der Diagnosekommunikation zu gewährleisten. Die Norm erleichtert es auch Originalherstellern (OEMs) und/oder Anbietern, diesen Dienst basierend auf ihren Sicherheitsanforderungen zu konfigurieren. Die Beispielkonfigurationen sind die unilaterale/bilaterale Authentifizierung, die PKI-basierte/Frage-Antwort-basierte Authentifizierung, der Aufbau eines sicheren Kanals, Sicherheitsalgorithmen und so weiter.

**[0049]** Bei solchen Konfigurationsparametern ist es zwingend erforderlich, einen sicheren Designprozess zu implementieren, um sicherzustellen, dass keine zusätzlichen Schwachstellen oder Schlupflöcher eingeführt werden. Eine solche Konfiguration, die eine Sicherheitslücke einführt, ist die Verwendung des Diffie-Hellman-Schlüsselvereinbarungsprotokolls mit unidirektionaler Authentifizierung. Wird Diffie-Hellmann als Schlüsselvereinbarungsprotokoll geplant ist, empfiehlt es sich, die bilaterale Authentifizierung als Mindestanforderung zu verwenden, um die Sicherheit des Dienstes 29 zu gewährleisten.

**[0050]** Die UDS-Norm ist bei allen Produkten aus verschiedenen Fahrzeugbereichen weit verbreitet. Produkte wie etwa Motorsteuergeräte, zentrale Gateways, Lenkungssteuergeräte sowie Testergeräte basieren auf der UDS-Norm. Das vorgeschlagene Verfahren kann bei jedem der vorstehend aufgeführten Produkte durchgeführt werden.

### Patentansprüche

1. Verfahren zur Authentifizierung eines Clients (100) gegenüber einem Server (102) für einen Diagnosedienst, wobei
  - der Client den Diagnosedienst durch Herstellen einer Verbindung mit einem Server (102) anfordert, indem er ein Zertifikat sendet, das den Client (100) identifiziert,
  - der Server die Anforderung des Clients (100) validiert und auf die Anforderung antwortet, indem er Informationen zum Aufbau eines sicheren Kanals an den Client (100) sendet,
  - der Client (100) die Antwort des Servers (102) validiert und den sicheren Kanal zum Server (102) aufbaut,
  - beim Einleiten der Verbindung mit dem Server (102) durch Senden eines Zertifikats, das den Client (100) identifiziert, eine Präambel zu einem Handshake verwendet wird, um das Problem eines zuvor abgefangenen Client-Zertifikats anzugehen.
2. Verfahren nach Anspruch 1, wobei das Verfahren zum Abschwächen eines Denial-of-Service-Angriffs auf den Server (102) zur Erstellung einer Frage dient.
3. Verfahren nach Anspruch 1 oder 2, wobei das Verfahren zum Abschwächen eines Denial-of-Service-Angriffs auf den Server (102) bezüglich eines Eigentumsnachweises verwendet wird.
4. Verfahren nach einem der Ansprüche 1 bis 3, wobei zur Bestimmung der Präambel ein Cookie berechnet wird.
5. Verfahren nach Anspruch 4, wobei das Cookie hauptsächlich als Hash mit zwei Eingaben berechnet wird.
6. Verfahren nach Anspruch 5, wobei die zwei Eingaben zur Hash-Berechnung ein ECU-Geheimnis und ein Zeitstempel sind.
7. Verfahren nach einem der Ansprüche 1 bis 6, ferner umfassend den Schritt der Authentifizierung des Servers (102) gegenüber dem Client (100), falls der Client (100) eine Server-Authentifizierung anfordert.
8. Verfahren nach einem der Ansprüche 1 bis 6, wobei ein nicht authentifiziertes Schlüsselvereinbarungsprotokoll verwendet wird, das Verfahren den Schritt der bilateralen Authentifizierung umfasst, um das Problem nicht authentifizierter kurzlebiger Schlüssel anzugehen.
9. Verfahren nach Anspruch 7, wobei das Verfahren zum Abschwächen eines Server-Spoof-Angriffs gegen den Client (100) verwendet wird.

10. Anordnung zur Authentifizierung, die geeignet ist, ein Verfahren nach einem der Ansprüche 1 bis 9 durchzuführen.

Es folgen 5 Seiten Zeichnungen

Anhängende Zeichnungen

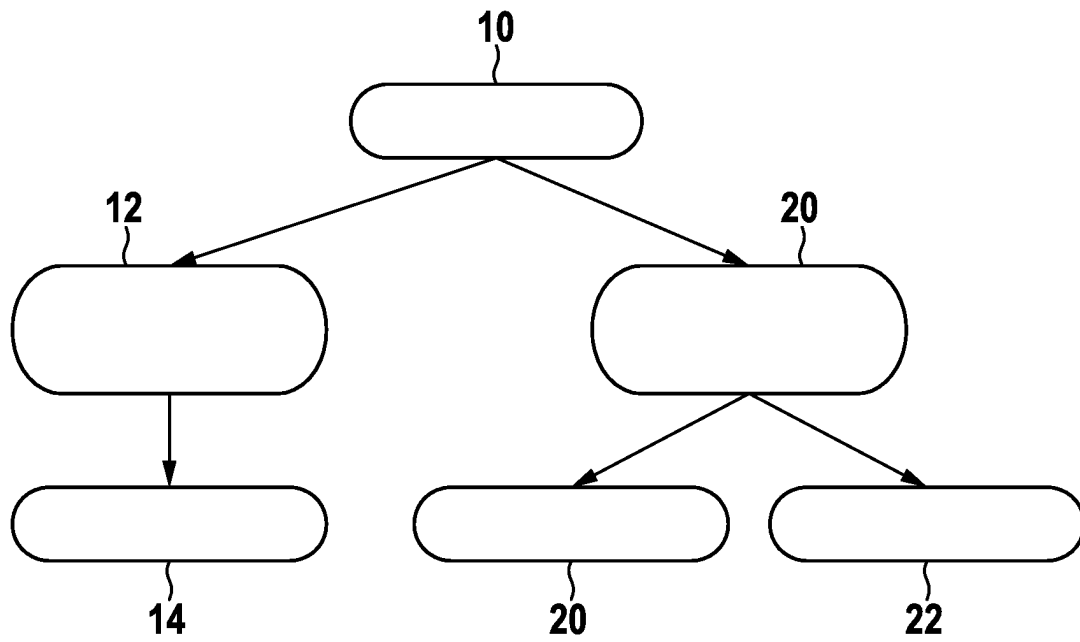
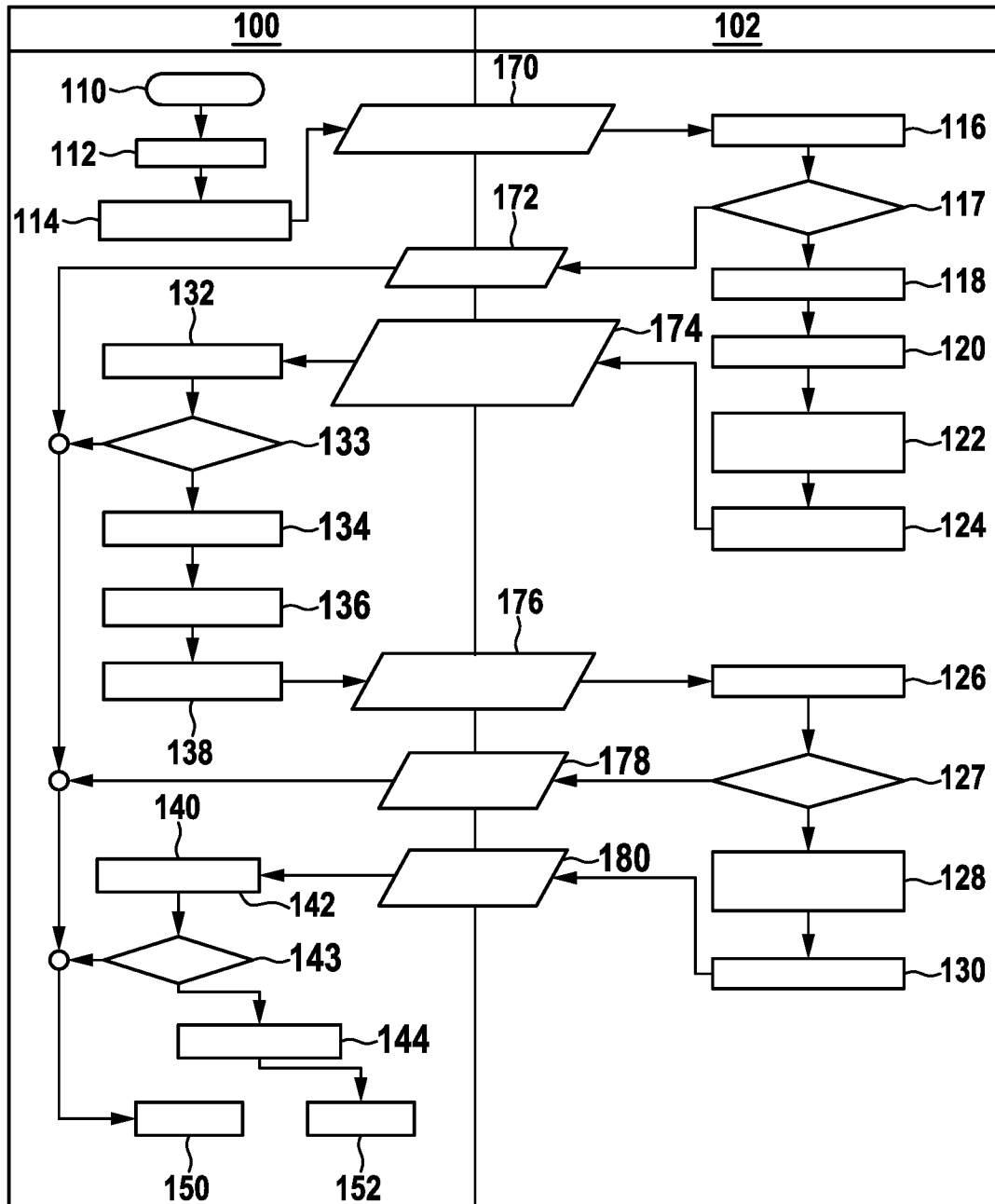


**Fig. 1**

**Fig. 2**

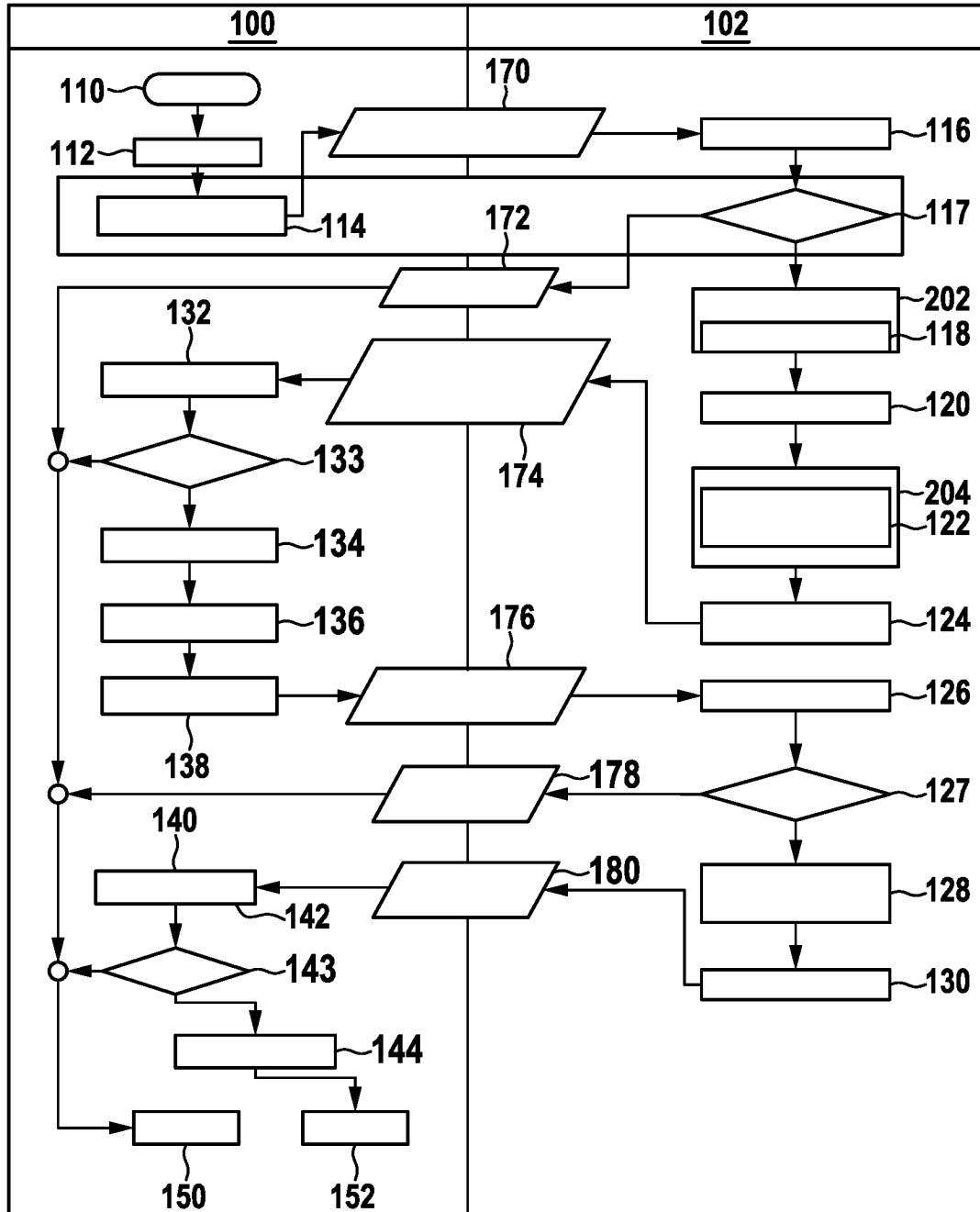


Fig. 3

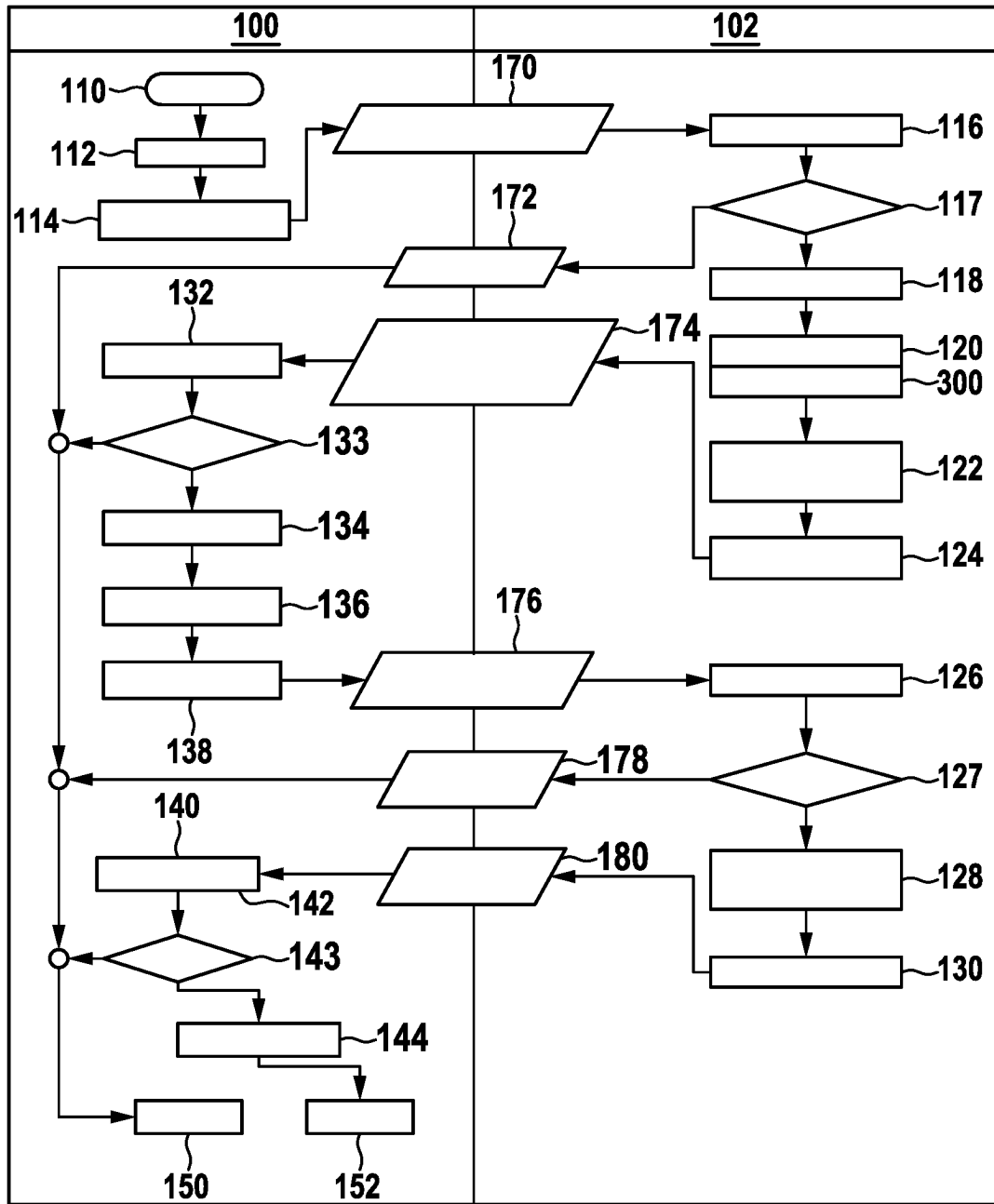


Fig. 4

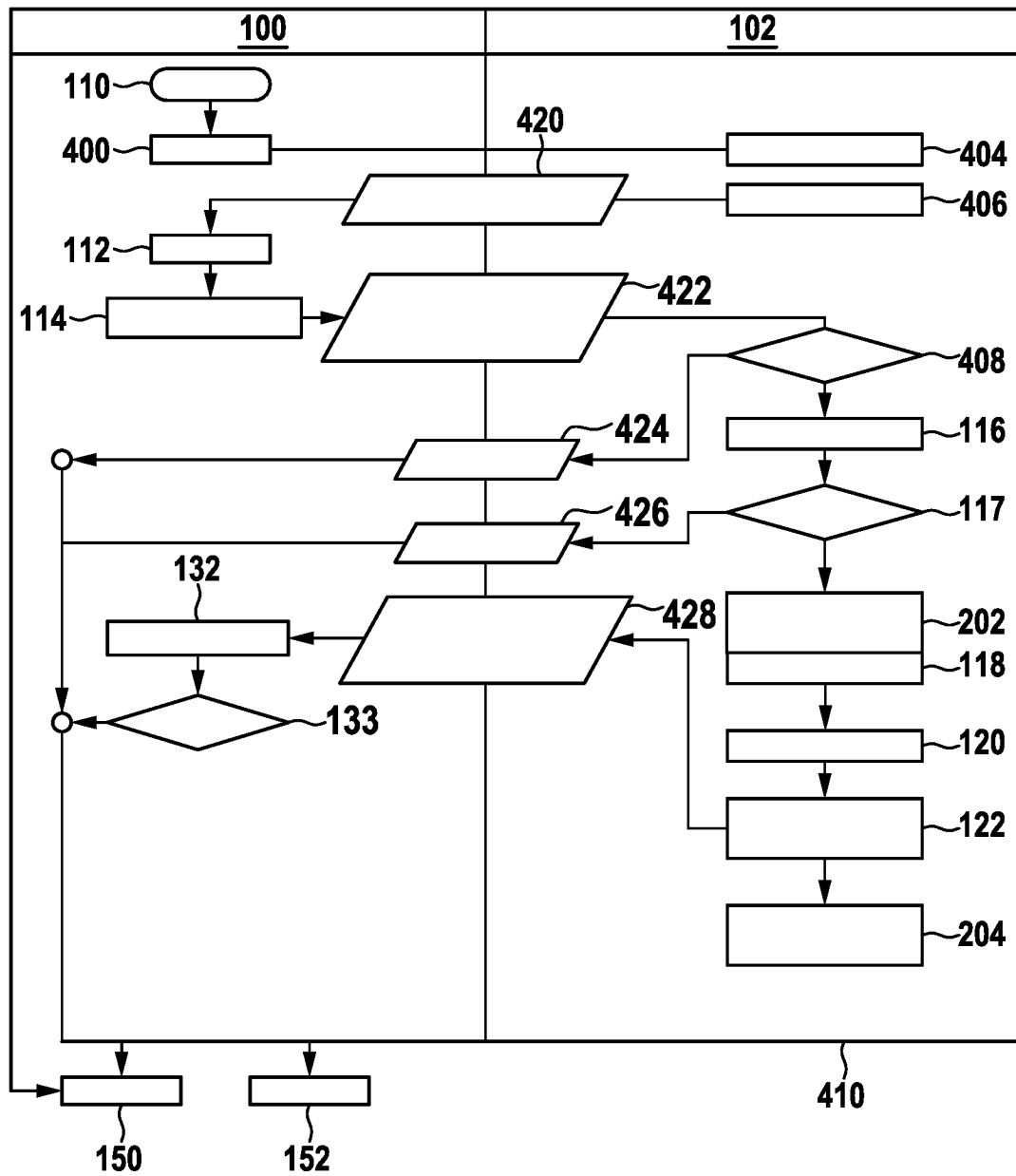


Fig. 5