



ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21)(22) Заявка: 2014118918/08, 31.10.2011

(24) Дата начала отсчета срока действия патента:
31.10.2011

Приоритет(ы):

(22) Дата подачи заявки: 31.10.2011

(43) Дата публикации заявки: 10.12.2015 Бюл. № 34

(45) Опубликовано: 27.04.2016 Бюл. № 12

(56) Список документов, цитированных в отчете о
поиске: US 2007/0234041 A1, 04.10.2007. US
2007/0021105 A1, 25.01.2007. US 2006/0236106
A1, 19.10.2006. WO 2010/095988 A1, 26.08.2010.
RU 101231 U1, 10.01.2011.(85) Дата начала рассмотрения заявки РСТ на
национальной фазе: 02.06.2014(86) Заявка РСТ:
FI 2011/050953 (31.10.2011)(87) Публикация заявки РСТ:
WO 2013/064716 (10.05.2013)

Адрес для переписки:

191036, Санкт-Петербург, а/я 24, "НЕВИНПАТ"

(72) Автор(ы):

ХОЛЬТМАННС Силке (FI),
ЛАЙТИНЕН Пекка Йоханнес (FI)

(73) Патентообладатель(и):

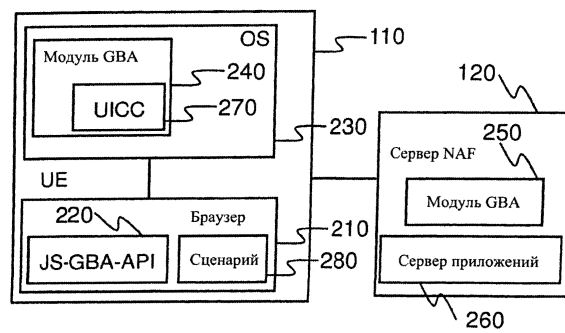
Нокиа Текнолоджиз Ой (FI)

(54) МЕХАНИЗМ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ДЛЯ ВНЕШНЕГО КОДА

(57) Реферат:

Изобретение относится к области обеспечения безопасности для внешнего кода, предоставляемого веб-сервером приложений. Техническим результатом является эффективное получение ключа начальной загрузки для внешнего кода. Способ реализации механизма обеспечения безопасности для внешнего кода содержит прием внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF); определение идентификатора сервера (NAF-Id) и генерацию специфического для сервера ключа начальной

загрузки (Ks_NAF) на основе идентификатора сервера (NAF-Id); определение маркера доступа; генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа и использование специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода. 6 н. и 14 з.п. ф-лы, 7 ил.



Фиг. 2



FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 582 863** (13) **C2**

(51) Int. Cl.

G06F 21/00 (2013.01)

H04L 29/06 (2006.01)

(12) ABSTRACT OF INVENTION

(21)(22) Application: **2014118918/08, 31.10.2011**

(24) Effective date for property rights:
31.10.2011

Priority:

(22) Date of filing: **31.10.2011**

(43) Application published: **10.12.2015** Bull. № 34

(45) Date of publication: **27.04.2016** Bull. № 12

(85) Commencement of national phase: **02.06.2014**

(86) PCT application:
FI 2011/050953 (31.10.2011)

(87) PCT publication:
WO 2013/064716 (10.05.2013)

Mail address:

191036, Sankt-Peterburg, a/ja 24, "NEVINPAT"

(72) Inventor(s):

HOLTMANN Silke (FI),

LAITINEN Pekka Johannes (FI)

(73) Proprietor(s):

Nokia Technologies Oy (FI)

(54) SECURITY MECHANISM FOR EXTERNAL CODE

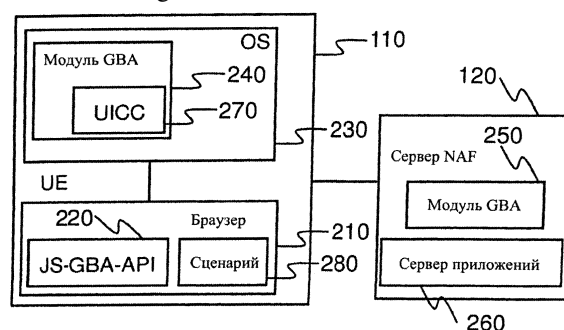
(57) Abstract:

FIELD: safety.

SUBSTANCE: invention relates to providing security for external code provided by web application server. Method of implementing security mechanism for external code comprises receiving external code containing a specific request to boot server key (Ks_NAF); determining server identifier (NAF-Id) and generation-specific boot server key (Ks_NAF), based on server identifier (NAF-Id); defining access token; generation of a specific code for external boot key (Ks_js_NAF) using a specific key for boot server (Ks_NAF) and a marker of access and use of a specific code for external boot key (Ks_js_NAF) to ensure security of external code mechanism.

EFFECT: technical result is efficient production of boot key for external code.

20 cl, 7 dwg



Фиг. 2

Настоящее изобретение в целом относится к механизму обеспечения безопасности для внешнего кода, предоставляемого веб-сервером приложений. В частности, хотя и не исключительно, настоящее изобретение относится к способу, посредством которого сервер, браузер, установленный в телефоне, и операционная система могут активизировать режим безопасного использования основанных на сотовой связи учетных данных пользователя из браузера с помощью внешнего кода, например кода JavaScript.

УРОВЕНЬ ТЕХНИКИ

В настоящее время доступно множество страниц на размещенных по всему миру веб-сайтах (например, HTML-документов), которые содержат различную текстовую и иную информацию. В то же время в настоящий момент наблюдается значительное распространение средств связи, в том числе беспроводных средств связи. Благодаря такому развитию технологических средств появилась возможность подключать небольшие переносные электронные устройства, такие как персональное информационное устройство (PDA, personal digital assistant), ноутбук, сотовый телефон, планшетный компьютер и другие электронные устройства, к тем же информационным источникам, например к веб-серверам или базам данных, к которым может подключаться персональный компьютер (PC, personal computer) и установленный на PC браузер. Существует несколько клиентских браузеров для небольших устройств, которые отображают содержимое веб-страниц на дисплее портативных устройств.

При этом в веб-контенте, таком как HTML-документ, используются команды сценариев, написанных на языке JavaScript или на подобном языке сценариев. Команды сценариев, выполняемые установленным на PC браузером, могут генерировать весь информационный контент или его часть, доступную пользователю установленного на PC браузера.

Новые мультимедийные мобильные терминалы (мультимедийные телефоны) поддерживают открытую платформу разработки для разработчиков приложений, позволяющую независимым разработчикам приложений формировать новые услуги и приложения для мультимедийной среды. В свою очередь, пользователи могут загружать новые приложения/услуги на свои мобильные терминалы и затем их использовать. Следовательно, для обеспечения полной безопасности важное значение приобретает процесс взаимодействия модуля обеспечения безопасности с веб-сервером приложений. Таким образом, требуется реализовать улучшенное решение по использованию модуля обеспечения безопасности мобильного терминала для веб-контента, содержащего внешний код, такой как JavaScript, считанный из внешнего источника.

СУЩНОСТЬ ИЗОБРЕТЕНИЯ

В соответствии с первым аспектом настоящего изобретения предлагается способ реализации механизма обеспечения безопасности для внешнего кода, включающий:

прием внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF);

определение идентификатора сервера (NAF-Id) и генерацию специфического для сервера ключа начальной загрузки (Ks_NAF) на основе идентификатора сервера (NAF-Id);

определение маркера доступа;

генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа и

использование специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

В варианте осуществления настоящего изобретения способ также включает определение маркера доступа (security token) с использованием первого произвольного запроса (RAND1, random challenge) и второго произвольного запроса (RAND2). Способ также может включать передачу второго произвольного запроса (RAND2) и специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) в сервер приложений для проверки достоверности специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF). Затем может передаваться ответный внешний код, содержащий второй произвольный запрос (RAND2) и специфический для внешнего кода ключ начальной загрузки (Ks_js_NAF).

В соответствии с вариантом осуществления настоящего изобретения способ также включает:

прием внешнего кода приложением браузера устройства из сервера приложений;
 определение идентификатора сервера (NAF-Id) и второго произвольного запроса (RAND2) прикладным программным интерфейсом (JS-GBA-API) браузера;
 запрос специфического для сервера ключа начальной загрузки (Ks_NAF) прикладным программным интерфейсом (JS-GBA-API) из модуля начальной загрузки операционной системы;
 прием специфического для сервера ключа начальной загрузки (Ks_NAF) прикладным программным интерфейсом (JS-GBA-API) из модуля начальной загрузки и генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) прикладным программным интерфейсом (JS-GBA-API).

Между приложением браузера устройства и сервером приложений может устанавливаться туннель безопасности на транспортном уровне (TLS, transport layer security). Идентификатор сервера (NAF-Id) может определяться с использованием доменного имени (FQDN) и идентификатора протокола обеспечения безопасности. Идентификатор протокола обеспечения безопасности может формироваться с использованием набора шифров обеспечения безопасности на транспортном уровне (TLS).

В варианте осуществления настоящего изобретения способ также включает генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с помощью функции формирования ключа. В состав внешнего кода может входить код JavaScript.

В варианте осуществления настоящего изобретения способ также включает определение маркера доступа с использованием главного ключа обеспечения безопасности на транспортном уровне (TLS).

В соответствии со вторым аспектом примера осуществления настоящего изобретения предлагается устройство, содержащее

по меньшей мере один процессор и по меньшей мере один модуль памяти, в котором хранится компьютерный программный код, при этом по меньшей мере один модуль памяти и компьютерный программный код сконфигурированы таким образом, чтобы при взаимодействии по меньшей мере с одним процессором устройство выполняло по меньшей мере следующие операции:

прием внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF);
 определение идентификатора сервера (NAF-Id) и генерация специфического для сервера ключа начальной загрузки (Ks_NAF) на основе идентификатора сервера (NAF-

Id);

определение маркера доступа;

генерация специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа и

использование специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

Маркер доступа может определяться с использованием первого произвольного запроса (RAND1) и второго произвольного запроса (RAND2).

В варианте осуществления настоящего изобретения по меньшей мере один модуль памяти и компьютерный программный код сконфигурированы таким образом, чтобы при взаимодействии по меньшей мере с одним процессором устройство выполняло по меньшей мере следующие операции:

прием внешнего кода приложением браузера устройства из сервера приложений;

определение идентификатора сервера (NAF-Id) прикладным программным интерфейсом (JS-GBA-API) браузера;

запрос специфического для сервера ключа начальной загрузки (Ks_NAF) прикладным программным интерфейсом (JS-GBA-API) из модуля начальной загрузки операционной системы;

прием специфического для сервера ключа начальной загрузки (Ks_NAF) прикладным программным интерфейсом (JS-GBA-API) из модуля начальной загрузки и

генерация специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) прикладным программным интерфейсом (JS-GBA-API).

Идентификатор сервера (NAF-Id) может определяться с использованием доменного имени (FQDN) и идентификатора протокола обеспечения безопасности.

В варианте осуществления настоящего изобретения маркер доступа может определяться с использованием главного ключа обеспечения безопасности на транспортном уровне (TLS).

Согласно третьему аспекту примера осуществления настоящего изобретения предлагается компьютерная программа, записанная на машиночитаемом носителе информации и содержащая программный код, при исполнении которого по меньшей мере одним процессором устройство выполняет следующие операции:

прием внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF);

определение идентификатора сервера (NAF-Id) и генерацию специфического для сервера ключа начальной загрузки (Ks_NAF) на основе идентификатора сервера (NAF-Id);

определение маркера доступа;

генерация специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа и

использование специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

В соответствии с четвертым аспектом примера осуществления настоящего изобретения предлагается способ реализации механизма обеспечения безопасности для внешнего кода, включающий:

передачу внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF);

определение маркера доступа;

генерацию специфического для сервера ключа начальной загрузки (Ks_NAF) с использованием идентификатора сервера (NAF-Id);

5 генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа и

использование специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

10 В соответствии с вариантом осуществления настоящего изобретения способ также включает:

запрос специфического для сервера ключа начальной загрузки (Ks_NAF) из функционального блока сервера начальной загрузки (BSF, bootstrapping server function);

определение идентификатора сервера (NAF-Id), содержащего доменное имя (FQDN) и идентификатор протокола обеспечения безопасности.

15 В соответствии с вариантом осуществления настоящего изобретения способ также включает:

прием специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) и проверку достоверности сгенерированного специфического для внешнего кода ключа

20 начальной загрузки (Ks_js_NAF) путем сравнения с принятым специфическим для внешнего кода ключом начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

В соответствии с пятым аспектом примера осуществления настоящего изобретения предлагается сервер приложений, содержащий:

по меньшей мере один процессор и

25 по меньшей мере один модуль памяти, в котором хранится компьютерный программный код, при этом по меньшей мере один модуль памяти и компьютерный программный код сконфигурированы таким образом, чтобы при взаимодействии по меньшей мере с одним процессором сервер приложений выполнял по меньшей мере следующие операции:

30 передача внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF);

генерация специфического для сервера ключа начальной загрузки (Ks_NAF) с использованием идентификатора сервера (NAF-Id);

определение маркера доступа;

35 генерация специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа и

использование специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

40 В варианте осуществления настоящего изобретения по меньшей мере один модуль памяти и компьютерный программный код сконфигурированы таким образом, чтобы при взаимодействии по меньшей мере с одним процессором сервер приложений выполнял по меньшей мере следующие операции:

45 генерация специфического для сервера ключа начальной загрузки (Ks_NAF) путем запроса из функционального блока сервера начальной загрузки (BSF).

Согласно шестому аспекту примера настоящего изобретения предлагается компьютерная программа, записанная на машинно-читаемом носителе информации и содержащая программный код, при исполнении которого по меньшей мере одним

процессором сервер приложений выполняет следующие операции:

передача внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF);

генерация специфического для сервера ключа начальной загрузки (Ks_NAF) с

5 использованием идентификатора сервера (NAF-Id);

определение маркера доступа;

генерация специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа и

10 использование специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

К любым упомянутым выше машиночитаемым носителям информации могут относиться такие носители цифровых данных как диск или дискета для хранения данных, оптическое запоминающее устройство, магнитное запоминающее устройство,

15 голографическое запоминающее устройство, оптомагнитное запоминающее устройство, память с фазовым переходом, резистивная оперативная память, магнитная оперативная память, память на твердых электролитах, сегнетоэлектрическая оперативная память, органическая память или полимерная память. Носитель информации может встраиваться в устройство только в качестве памяти для хранения данных, или он может

20 формироваться в виде устройства, выполняющего другие функции, включая, помимо прочего, функции памяти компьютера, микросхемы, или подсистемы электронного устройства.

Выше были представлены различные частные примеры аспектов и вариантов осуществления настоящего изобретения. Приведенные выше варианты осуществления настоящего изобретения используются только для разъяснения аспектов или шагов, которые могут использоваться при реализации настоящего изобретения. Некоторые варианты осуществления могут быть представлены только со ссылкой на определенные аспекты изобретения. Следует принимать во внимание, что соответствующие варианты осуществления могут также применяться к другим аспектам, представленным в качестве

30 примера.

КРАТКОЕ ОПИСАНИЕ ЧЕРТЕЖЕЙ

Настоящее изобретение описывается посредством приводимых примеров со ссылкой на прилагаемые чертежи, на которых:

на фиг. 1 показаны некоторые компоненты архитектуры системы, в которой могут применяться различные варианты осуществления настоящего изобретения;

на фиг. 2 показаны некоторые компоненты элементов системы, в которой могут применяться различные варианты осуществления настоящего изобретения;

на фиг. 3 показана схема обмена сообщениями в соответствии с вариантом осуществления настоящего изобретения;

40 на фиг. 4 показан пример блок-схемы сервера приложений, в котором могут применяться различные варианты осуществления настоящего изобретения;

на фиг. 5 показан пример блок-схемы пользовательского устройства, в котором могут применяться различные варианты осуществления настоящего изобретения;

на фиг. 6 показан алгоритм выполнения операций в пользовательском устройстве в соответствии с вариантом осуществления настоящего изобретения, и

на фиг. 7 показан алгоритм выполнения операций в сервере приложений в соответствии с вариантом осуществления настоящего изобретения.

ПОДРОБНОЕ ОПИСАНИЕ

В последующем описании одинаковым номерам соответствуют одни и те же элементы.

В различных вариантах осуществления настоящего изобретения используются положения общей архитектуры аутентификации (GAA, Generic Authentication Architecture) и общей архитектуры начальной загрузки (GBA, Generic Bootstrapping Architecture),
 5 определенные в стандартах 3GPP для аутентификации равноправных объектов и обеспечения безопасности при связи. Варианты GAA/GBA стандартизованы открытым сообществом производителей мобильной связи (OMA, OMA) и консорциумом CableLabs. Принципы GAA/GBA основаны на алгоритмах АКА (Authentication and Key Agreement, аутентификация и соглашения о ключах) мобильной связи для 3GPP. Исходная цель
 10 процедур GAA/GBA заключается в аутентификации пользовательского оборудования или абонента. В различных вариантах осуществления настоящего изобретения процедуры GAA/GBA используются для повышения уровня безопасности при взаимодействии сервера приложений, браузера мобильного терминала и операционной системы мобильного терминала. В варианте осуществления настоящего изобретения
 15 использование учетных данных пользователя, основанных на сотовой связи, осуществляется из браузера с помощью внешнего кода, загружаемого из сервера в пользовательское устройство, например, на веб-странице. В состав внешнего кода может входить, например, код JavaScript.

Преимущество, которое можно достичь с помощью механизмов аутентификации
 20 GAA/GBA, применяемых для аутентификации серверов приложений, заключается в том, что можно не прибегать к использованию дорогостоящей инфраструктуры публичных ключей, которая необходима в противном случае. GAA/GBA является многоцелевым активизатором, используемым, например, для передвижных телевизионных систем и услуг определения присутствия. С помощью этого механизма
 25 и связанной с ним инфраструктуры можно снизить административные расходы и объем требуемых инвестиций.

В приведенном ниже описании такие термины как пользовательское устройство, сервер приложений, сервер начальной загрузки и общая процедура аутентификации применяются для обозначения различных элементов/механизмов, относящихся к
 30 различным вариантам осуществления настоящего изобретения. Следует отметить, что помимо или вместо этих явно указанных элементов/механизмов могут использоваться некоторые другие элементы/механизмы, обеспечивающие сопоставимую функциональность.

В различных вариантах осуществления настоящего изобретения сервер приложений
 35 может представлять собой веб-сервер, предоставляющий веб-услугу пользователю. Кроме того, сервер приложений может быть не защищен сетевым оператором и может включать в свой состав функциональный блок сетевых приложений (NAF, network application function).

На фиг. 1 показаны некоторые компоненты архитектуры системы 100, в которой
 40 могут применяться различные варианты осуществления настоящего изобретения; Система содержит пользовательское устройство, такое как пользовательское оборудование (UE, user equipment) 110, и сервер 120 приложений, предоставляющий веб-услугу(-и). Кроме того, система включает функциональный блок 130 сервера начальной загрузки (BSF) и абонентскую базу 140 данных, такую как опорный
 45 абонентский сервер (HSS, home subscriber server) или опорный регистр местоположения (HLR, home location register). Устройство 110 также содержит функциональный блок 150 GBA, сконфигурированный для совместной работы с функциональным блоком сервера начальной загрузки (BSF), и клиента функционального блока 160 сетевых приложений

(NAF), сконфигурированного для совместной работы с сервером 120 приложений. Клиент функционального блока сетевых приложений (NAF) может включать в свой состав, например, браузер. Согласно GAA/GBA сервер приложений может называться функциональным блоком сетевых приложений (NAF). Между функциональным блоком 130 сервера начальной загрузки (BSF) и абонентской базой 140 данных в GAA/GBA 5 определен интерфейс Zh, между функциональным блоком 130 сервера начальной загрузки (BSF) и сервером 120 приложений определен интерфейс Zn, а между функциональным блоком 130 сервера начальной загрузки (BSF) и пользовательским устройством 110 определен интерфейс Ub. Кроме того, если установлен HLR, может 10 быть реализован интерфейс Zh' между функциональным блоком 130 сервера начальной загрузки (BSF) и HLR 140. Помимо этого, может существовать интерфейс с функциональным блоком локатора места подписки (SLF, subscription locator function), который предоставляет функциональному блоку сервера начальной загрузки (BSF) информацию из абонентской базы данных.

Административное управление сервером 120 приложений может выполняться 15 стороной, отличной от той, что управляет функциональным блоком 130 сервера начальной загрузки (BSF) и абонентской базой 140 данных, либо управление этими устройствами может выполняться одной стороной (обычно - оператором рассматриваемой сети связи).

20 Следует отметить, что система 100 может также содержать, и обычно содержит, другие различные элементы, не показанные на чертеже, чтобы не усложнять описание.

В варианте осуществления настоящего изобретения общий функциональный блок 130 сервера начальной загрузки (BSF) и пользовательское оборудование (UE) 110 25 выполняют совместную аутентификацию с использованием протокола аутентификации и принятия соглашения о ключах (AKA), после чего соглашение о ключах применяется при взаимодействии пользовательского оборудования (UE) 110 и функционального блока сетевых приложений (NAF) сервера 120. Функциональный блок сетевых приложений (NAF) является функциональным модулем, размещенным в сервере 120, предоставляющем услуги. В альтернативном варианте, если используется традиционная 30 смарт-карта, могут использоваться функции обеспечения безопасности на транспортном уровне (TLS, transport layer security) и традиционная аутентификация. В рамках общей архитектуры начальной загрузки (GBA) может также использоваться другой механизм аутентификации, такой как хэш протокола передачи гипертекста (HTTP, hypertext transfer protocol) или хэш протокола инициирования сеансов (SIP, session initiation protocol). К 35 основным задачам функционального блока сетевых приложений (NAF) сервера 120 относятся управление услугами/пользователями (например, подписка и отмена подписки на услугу) и управление ключами услуг (например, генерация и доставка ключа услуги). Функциональный блок 130 сервера начальной загрузки должен ограничивать сферу применения набора ключей конкретным функциональным блоком сетевых приложений 40 (NAF) сервера 120 путем использования процедуры формирования ключей. Процедура формирования ключей может использоваться с множеством функциональных блоков сетевых приложений (NAF) в течение срока действия набора ключей. Срок действия набора ключей устанавливается в соответствии с локальной политикой функционального блока 130 сервера начальной загрузки (BSF). Для функционального блока 130 сервера 45 начальной загрузки (BSF) допускается осуществлять выборку любой требуемой информации об аутентификации, информации о безопасности и профиле абонента из опорной абонентской системы (HSS, home subscriber system) 140. В существующих сетях функциональный блок 130 сервера начальной загрузки (BSF) может взаимодействовать

с опорным регистром местоположения (HLR), а не с опорной абонентской системой (HSS) 140.

С точки зрения, например, операторов применение GBA на основе внешнего кода, такого как код сценария, написанного, например, на языке JavaScript, может вызывать определенную обеспокоенность. Внешний код может загружаться в пользовательское оборудование, и обеспокоенность может быть вызвана тем, что секретная информация модуля GBA передается в веб-сервер в исходном виде.

На фиг. 2 показаны некоторые компоненты элементов системы, в которой могут применяться различные варианты осуществления настоящего изобретения. Внешний код может включать любой код, загружаемый в устройство и потенциально используемый или исполняемый локально. Внешний код может выполняться установленными приложениями, такими, например, как браузеры или мини-приложения. Одним из примеров внешнего кода является код JavaScript. Для простоты следующие примеры осуществления настоящего изобретения описываются с использованием JavaScript, однако варианты осуществления не ограничены кодом JavaScript, в этих вариантах может применяться любой внешний код.

JavaScript может использоваться в виде кода JavaScript, обрабатываемого на стороне клиента в пользовательском оборудовании (UE) 110. Выполнение кода 280 JavaScript может быть реализовано в среде веб-браузера 210 с целью поддержки улучшенных пользовательских интерфейсов и динамических веб-сайтов. Это позволяет выполнять программный доступ к вычислительным объектам в среде центральной машины. Код 280 JavaScript может также использоваться приложениями вне веб-страниц, например, в документах, в браузерах, специфических для сайтов, и в виджетах настольных компьютеров. JavaScript также используется для веб-приложений на стороне сервера. Прикладной программный интерфейс (API, application programming interface) представляет собой конкретный набор правил ('код') и спецификаций, которым могут следовать программы для связи друг с другом. API служит в качестве интерфейса между различными программами и облегчает процесс взаимодействия между ними. Для кода 280 JavaScript может создаваться интерфейс GBA API, обозначаемый на фиг. 2 как JS-GBA-API 220. Операционная система (OS, operating system) 230 пользовательского оборудования (UE) 110 может включать в свой состав модуль 240 GBA, отвечающий за управление безопасностью пользовательского оборудования (UE) 110.

Пользовательское оборудование (UE) 110 может также содержать универсальную карту 270 с интегральной схемой (UICC, universal integrated circuit card), представляющую собой смарт-карту, используемую в мобильных терминалах в сетях сотовой связи.

Универсальная карта 270 с интегральной схемой (UICC) обеспечивает целостность и безопасность всех видов персональных данных, и на ней обычно записаны приложения. Смарт-карта UICC может также содержать CPU, ROM, RAM, EEPROM и схемы ввода/вывода.

Браузер 210 пользовательского оборудования (UE) 110 может осуществлять связь с функциональным блоком сетевых приложений (NAF) сервера 120, работающего в качестве сервера прикладных услуг, например, для веб-контента.

Функциональный блок сетевых приложений (NAF) сервера 120 может содержать, например, модуль 250 GBA NAF и сервер 260 приложений.

В варианте осуществления настоящего изобретения обеспечивается взаимодействие модуля управления безопасностью пользовательского устройства (модуля GBA, являющегося частью OS) с сервером веб-приложений. Механизм обеспечения безопасности позволяет безопасно использовать модуль 240 управления безопасностью

из браузера 210 с помощью кода 280 JavaScript, поступающего из внешнего источника 120.

При упоминании ключей GBA подразумеваются следующие ключи: Ks и специфические для NAF ключи, сформированные на основе Ks. При упоминании специфических для NAF ключей подразумеваются следующие ключи: Ks_ext/int_NAF (в контексте GBAJJ) и Ks_NAF (в контексте GBA_ME), а также любые ключи, сформированные на основе этих ключей. Ks_ext_NAF представляет собой тот же ключ, что и Ks_NAF, то есть специфический для NAF ключ, используемый в ME. Ключ Ks_ext_NAF формируется на карте UICC в контексте GBA_U и передается в ME, а ключ Ks_NAF формируется в ME в контексте GBA_ME. Оба этих ключа используются в ME одинаковым образом, независимо от контекста. Ks_int_NAF формируется и используется в UICC. Ks_int_NAF никогда не передается из UICC. При ссылке на ключ Ks_js_NAF подразумевается ключ JavaScript для кода JavaScript и сервера приложений, используемый вместо Ks_NAF или Ks_ext_NAF.

Согласно варианту осуществления настоящего изобретения, перед началом связи между UE и функциональным блоком сетевых приложений (NAF) эти два устройства вначале должны прийти к соглашению о том, требуется ли использовать GBA. Если устройству UE требуется взаимодействовать с функциональным блоком сетевых приложений (NAF), но при этом отсутствует информация о том, требуется ли функциональному блоку сетевых приложений (NAF) использовать общедоступные ключи, полученные посредством GBA, устройство UE обращается к функциональному блоку сетевых приложений (NAF) для получения последующих инструкций.

На фиг. 3 показана схема обмена сообщениями в соответствии с вариантом осуществления настоящего изобретения. Не все показанные сообщения и шаги должны выполняться, порядок передачи сообщений может изменяться, и количество сообщений может увеличиваться, без ограничения сообщениями и шагами, показанными на фиг. 3.

Пользовательское устройство, такое как пользовательское оборудование (UE), может через опорную точку Ua начать связь с сервером приложений, таким как сервер функционального блока сетевых приложений (NAF), без использования каких-либо параметров, связанных с общей архитектурой начальной загрузки (GBA). Если блоку NAF требуется использовать общие ключи, полученные посредством GBA, но запрос, поступивший из UE, не содержит параметры, связанные с GBA, функциональный блок сетевых приложений (NAF) передает ответное сообщение инициирования начальной загрузки. Формат такой индикации может зависеть от конкретной опорной точки Ua.

Согласно варианту осуществления настоящего изобретения, веб-браузер 210 рассматривается как защищенное приложение с точки зрения доверительного отношения пользователя к браузеру 210 в том, что касается корректного выполнения функций, связанных с обеспечением безопасности, и предотвращения утечки к сторонним пользователям существенной информации, такой как пароли. На фиг. 3 показано, что веб-браузер 210 разделен на три функциональных блока: модуль 310 ядра, модуль 320 JavaScript и модуль 330 GBA-API.

Модуль 310 ядра выполняет основные функции для веб-браузера 210, такие как установка параметров обеспечения безопасности на транспортном уровне (TLS) при взаимодействии с веб-серверами 120, загрузка веб-ресурсов и предоставление пользователю информации о пользовательском интерфейсе.

Модуль 330 GBA API обеспечивает прикладной программный интерфейс (API) с любым кодом JavaScript, выполняющимся в веб-браузере 210. Поскольку в коде JavaScript

не должны применяться явные средства защиты, веб-браузер 210 и GBA API 330 не предоставляют какой-либо существенной информации модулю JavaScript, а также принимают из модуля JavaScript только необходимую существенную информацию.

Модуль 320 JavaScript выполняет загруженный сценарий JavaScript. Любой код JavaScript, выполняемый веб-браузером 210, следует рассматривать как незащищенный, при этом доступ к существенным ресурсам при выполнении такого кода не должен предоставляться либо должен контролироваться.

Показанный на фиг. 3 алгоритм может выполняться в рамках обеспечения безопасности на транспортном уровне (TLS) с серверной аутентификацией. Кроме того, веб-браузер 210 может выполнять загрузку html-страницы, в которой один из связанных с ней ресурсов JavaScript называется "gba.js".

На шаге 0 приложение 210 браузера и веб-сервер 120 устанавливают туннель обеспечения безопасности на транспортном уровне (TLS), аутентифицированный сервером.

На шаге 1, показанном на фиг. 3, приложение 210 браузера пользовательского устройства, такого как пользовательское оборудование (UE), запрашивает загрузку контента. Контент может представлять собой, например, веб-страницу, предоставляемую сервером 120 приложений, таким как веб-сервер. Запрос, передаваемый на шаге 1, может содержать, например, HTTP-запрос.

На шаге 2, показанном на фиг. 3, веб-сервер 120 динамически создает файл "gba.js" с кодом JavaScript путем генерации произвольного серверного запроса (RAND1), который подлежит включению в код JavaScript и предоставляется интерфейсу 330 GBA API браузера 210. RAND1 также локально сохраняется на веб-сервере 120. В этом коде JavaScript прикладной программный интерфейс (API) 220 может использоваться для запроса и получения специфического для JavaScript ключа GBA (Ks_js_NAF). Произвольный запрос RAND1 включается на шаге 2 в запрос GBA API. Запрос специфического для JavaScript ключа GBA (Ks_js_NAF) может также пересылаться в модуль 240 GBA после приема браузером 210 и может пересылаться модулем 240 GBA в GBA API 220 для последующей обработки.

На шаге 3 из сервера 120 загружается веб-страница с кодом 280 JavaScript в виде, например, HTTP-ответа. На шаге 4 запускается ядро 310 веб-браузера 210 для выполнения кода "gba.js", написанного на JavaScript, в модуле 320 JavaScript.

На шаге 5 код JavaScript "gba.js" выполняется до точки, в которой вызывается GBA API 330. При вызове в качестве одного из параметров передается RAND1. На шаге 6 JavaScript GBA API 330 сохраняет принятое значение RAND1. GBA API 330 также находит соответствующую информацию о коде JavaScript, например о том, какая html-страница обрабатывается, с какого url загружена html-страница, и какой набор ключей TLS используется в туннеле TLS. Доменное имя (FQDN) веб-сервера (NAF) 120 может извлекаться из url веб-страницы, и идентификатор протокола обеспечения безопасности в точке Ua может формироваться на основе используемого набора ключей TLS. Доменное имя (FQDN) сервера 120 NAF и идентификатор протокола обеспечения безопасности в точке Ua формируют идентификатор функционального блока сетевых приложений (NAF-Id, network application function identifier).

На шаге 7 модуль 330 GBA API передает вызов в модуль 240 GBA с использованием идентификатора NAF-Id, полученного на шаге 6. На шаге 8 модуль 240 GBA выполняет начальную загрузку с помощью функционального блока начальной загрузки (BSF), в том случае если отсутствует действительный главный ключ Ks GBA. На основе Ks формируется специфический для NAF ключ (Ks_ext_NAF) с использованием NAF-Id.

В случае GBA_ME карта UICC 270 передает СК и ИК в модуль 240 GBA, который на их основе генерирует Ks путем, например, конкатенации СК и ИК. Кроме того, модуль 240 GBA генерирует Ks_NAF с использованием Ks NAF-Id.

В случае GBAJJ карта UICC 270 непосредственно сохраняет СК и ИК и генерирует Ks_ext_NAF, который затем передается в модуль 240 GBA.

Таким образом, в случае GBA_ME все специфические для GBA функции реализуются в ME, а в случае GBA_U часть функций GBA реализуется в UICC 270. В основном, Ks хранится в UICC 270, и только полученный Ks_(ext)_NAF передается в модуль 240 GBA. Другими словами, "главный ключ" Ks для GBA генерируется либо в ME (в случае GBA_ME), либо в UICC 270 (в случае GBAJJ).

Приложение, получающее требуемый ключ GBA, взаимодействует только с модулем 240 GBA, и ключ GBA является либо ключом Ks_NAF в случае GBA_ME, либо ключом Ks_ext_NAF в случае GBA_U, соответственно. Приложение затем может использовать ключ GBA Ks_(ext)_NAF независимо от источника.

На шаге 9 модуль 240 GBA возвращает специфический для NAF ключ (Ks_ext_NAF) в GBA API 330 браузера, например, с идентификатором транзакции начальной загрузки (B-TID, bootstrapping transaction identifier) и сроком действия ключа. На шаге 10 интерфейс GBA API 330 может генерировать произвольный запрос RAND2 на стороне клиента. Маркер доступа может определяться с использованием произвольного запроса RAND1 и произвольного запроса RAND2. Кроме того, специфический для JavaScript ключ GBA (Ks_js_NAF) создается с использованием специфического для сервера ключа начальной загрузки (Ks_ext_NAF) и маркера доступа (произвольные запросы RAND1 и RAND2). Функция получения ключа (KDF, key derivation function) может использоваться для формирования специфического для JavaScript ключа GBA следующим образом:

$Ks_js_NAF = KDF(Ks_ext_NAF, RAND1 \parallel RAND2)$

RAND1 представляет собой произвольный запрос, принятый от сервера 120, а RAND2 генерируется средствами GBA API 330. Ks_(ext)_NAF может обрабатываться для GBA API 330 на уровне JavaScript. При этом может вызываться функция JavaScript, например GBA.getNAFKey(RAND1), которая возвращает значения Ks_js_NAF и RAND2.

На шаге 11 GBA API 330 возвращает выполняющемуся модулю 320 JavaScript специфический для JavaScript ключ Ks_js_NAF, RAND2, B-TID и срок действия ключа. На шаге 12 модуль 320 JavaScript продолжает выполняться и использует ключ Ks_js_NAF в соответствии с инструкциями, полученными от веб-сервера 120 (с использованием кода JavaScript "gba.js").

На шаге 13, после выполнения логических операций на стороне клиента, модуль 320 JavaScript передает запрос (например, HTTP-запрос) веб-серверу 120. Этот запрос может содержать, по меньшей мере, значения Ks_js_NAF, RAND2 и B-TID.

На шаге 14 веб-сервер 120 может извлечь Ks_ext_NAF из функционального блока начальной загрузки (BSF) и затем сформировать Ks_js_NAF с использованием принятого значения RAND2 и сохраненного значения RAND1. Веб-сервер 120 для проверки достоверности может сравнить принятое значение Ks_js_NAF со значением, сформированным локально. Если принятое значение Ks_js_NAF достоверно, веб-сервер 120 продолжает обработку запроса, выполненного на шаге 13, и на шаге 15 возвращает результат в модуль 320 JavaScript веб-браузера 120. Кроме того, веб-сервер 120 может продолжать выполнение кода JavaScript.

В варианте осуществления настоящего изобретения специфический для NAF ключ (Ks_NAF) не передается как таковой в сервер, что повышает уровень безопасности, обеспечиваемый таким механизмом. Кроме того, специфический для JavaScript ключ

(Ks_js_NAF) изменяется каждый раз при использовании GBA API 330, поскольку при этом изменяются значения RAND1 и RAND2. Такой механизм обеспечивает, например, дополнительную безопасность и защиту от воспроизведения.

В другом варианте осуществления настоящего изобретения используется другой маркер доступа. В таком варианте осуществления настоящего изобретения на шаге 2, показанном на фиг. 3, веб-сервер 120 выбирает файл с кодом "gba.js" на языке JavaScript, предоставляемый интерфейсу 330 GBA API браузера 210. В этом коде JavaScript прикладной программный интерфейс (API) 220 может использоваться для запроса и получения специфического для JavaScript ключа GBA (Ks_js_NAF). Запрос специфического для JavaScript ключа GBA (Ks_js_NAF) может также пересылаться в модуль 240 GBA после приема браузером 210 и может пересылаться модулем 240 GBA в GBA API 220 для последующей обработки.

На шаге 3 с сервера 120 загружается веб-страница с кодом 280 JavaScript в виде, например, HTTP-ответа. На шаге 4 запускается ядро 310 веб-браузера 210 для выполнения кода "gba.js", написанного на JavaScript, в модуле 320 JavaScript.

На шаге 5 код JavaScript "gba.js" выполняется до точки, в которой вызывается GBA API 330. На шаге 6 JavaScript GBA API 330 находит соответствующую информацию о коде JavaScript, например о том, какая html-страница обрабатывается, с какого url загружена html-страница, и какой набор ключей обеспечения безопасности на транспортном уровне (TLS) используется в туннеле TLS. Доменное имя (FQDN) веб-сервера (NAF) 120 может извлекаться из url веб-страницы, и идентификатор протокола обеспечения безопасности в точке Ua может вычисляться на основе используемого набора ключей TLS. Доменное имя (FQDN) NAF-сервера 120 и идентификатор протокола обеспечения безопасности в точке Ua формируют идентификатор функционального блока сетевых приложений (NAF-Id).

На шаге 7 модуль 330 GBA API передает вызов в модуль 240 GBA с использованием идентификатора NAF-Id, полученного на шаге 6. На шаге 8 модуль 240 GBA выполняет начальную загрузку с помощью функционального блока начальной загрузки (BSF), если отсутствует действительный главный ключ Ks GBA. На основе Ks формируется специфический для NAF ключ (Ks_ext_NAF) с использованием NAF-Id.

В случае GBA_ME карта UICC 270 передает СК и ИК в модуль 240 GBA, который на их основе генерирует Ks путем, например, конкатенации СК и ИК. Кроме того, модуль 240 GBA генерирует Ks_NAF с использованием Ks NAF-Id.

В случае GBA_U карта UICC 270 непосредственно сохраняет СК и ИК и генерирует Ks_ext_NAF, который затем передается в модуль 240 GBA.

Так, в случае GBA_ME все специфические для GBA функции реализуются в ME, а в случае GBA_U часть функций GBA реализуется в UICC 270. В основном, Ks хранится в UICC 270, и только полученный Ks_(ext)_NAF передается в модуль 240 GBA. Другими словами, "главный ключ" Ks для GBA генерируется либо в ME (в случае GBA_ME), либо в UICC 270 (в случае GBA_U).

Приложение, получающее требуемый ключ GBA, взаимодействует только с модулем 240 GBA, и ключ GBA является либо ключом Ks_NAF в случае GBA_ME, либо ключом Ks_ext_NAF в случае GBA_U, соответственно. Приложение затем может использовать ключ GBA Ks_(ext)_NAF независимо от источника.

На шаге 9 модуль 240 GBA возвращает специфический для NAF ключ (Ks_(ext)_NAF) в GBA API 330 браузера, например, с идентификатором транзакции начальной загрузки (B-TID) и сроком действия ключа. На шаге 10 после приема ключа Ks_(ext)_NAF интерфейс GBA API 330 браузера может определить маркер доступа. Маркер доступа

(TLS_MK_Extr) может извлекаться из главного ключа обеспечения безопасности на транспортном уровне (TLS) с использованием экспортируемой функции. Экспортируемой функции может, например, присваиваться метка "EXPORTER_3GPP_GBA_WEB". Маркер доступа (TLS_MK_Extr) может применяться для формирования специфического для JavaScript ключа Ks_js_NAF, который связан с туннелем TLS, аутентифицируемым сервером. Ks_js_NAF может формироваться на основе Ks_(ext)_NAF следующим образом:

$$Ks_js_NAF = KDF(Ks_(\text{ext})_NAF, TLS_MK_Extr)$$

Специфический для JavaScript ключ GBA (Ks_js_NAF) создается с использованием специфического для сервера ключа начальной загрузки (Ks_(ext)_NAF) и маркера доступа (TLS_ML_Extr). Функция формирования ключа (KDF) может использоваться для формирования специфического для JavaScript ключа GBA. Ks_(ext)_NAF может обрабатываться для GBA API 330 на уровне JavaScript.

На шаге 11 GBA API 330 возвращает выполняющемуся модулю 320 JavaScript специфический для JavaScript ключ Ks_js_NAF, B-TID и срок действия ключа. На шаге 12 модуль 320 JavaScript продолжает выполняться и использует ключ Ks_js_NAF в соответствии с инструкциями, полученными от веб-сервера 120 (с использованием кода "gba.js", написанного на JavaScript).

На шаге 13, после выполнения логических операций на стороне клиента, модуль 320 JavaScript передает запрос (например, HTTP-запрос) веб-серверу 120. Этот запрос может содержать, по меньшей мере, значения Ks_js_NAF и B-TID.

На шаге 14 веб-сервер 120 может извлечь Ks_(ext)_NAF из функционального блока начальной загрузки (BSF) и определить маркер доступа (TLS_MK_Extr) таким же образом, как и на шаге 10. Веб-сервер 120 затем может сформировать Ks_js_NAF с использованием маркера доступа (TLS_MK_Extr). Веб-сервер 120 для проверки достоверности может сравнить принятое значение Ks_js_NAF со значением, сформированным локально. Если принятое значение Ks_js_NAF достоверно, веб-сервер 120 продолжает обработку запроса, выполненного на шаге 13, и на шаге 15 возвращает результат в модуль 320 JavaScript веб-браузера 120. Кроме того, веб-сервер 120 может продолжать выполнение кода JavaScript.

В варианте осуществления настоящего изобретения специфический для NAF ключ (Ks_NAF) не передается как таковой в сервер, что повышает уровень безопасности, обеспечиваемый таким механизмом.

На фиг. 4 показан пример блок-схемы сервера 400 приложений, в котором могут применяться различные варианты осуществления настоящего изобретения. Это может быть веб-сервер, сервер загрузки файлов или любой сервер предоставления контента.

Общая структура сервера 400 приложений включает модуль 450 интерфейса связи, процессор 410, связанный с модулем 450 интерфейса связи, и память 420, связанную с процессором 410. Устройство также содержит программное обеспечение 430, хранимое в памяти 420 и предназначенное для загрузки в процессор 410 для выполнения.

Программное обеспечение 430 может состоять из одного или более программных модулей и может быть реализовано в виде компьютерного программного изделия.

Модуль 450 интерфейса связи реализует по меньшей мере часть процесса передачи данных, обсуждавшегося в связи с различными вариантами осуществления настоящего изобретения. Модуль 450 интерфейса связи может, например, представлять собой модуль радиоинтерфейса, такой как модуль радиосвязи в соответствии с технологиями WLAN, Bluetooth, GSM/GPRS, CDMA, WCDMA или LTE (Long Term Evolution, технология долгосрочного развития). Модуль 450 интерфейса связи может интегрироваться в сервер 400 приложений или в адаптер, плату и т.п., которые могут устанавливаться в

подходящем слоте или порте сервера 400 приложений. Модуль 450 интерфейса связи может поддерживать одну или множество технологий радиointерфейса. На фиг. 4 показан один модуль 450 интерфейса связи, однако сервер 400 приложений может содержать множество модулей интерфейсов связи. Модуль 450 интерфейса связи

5 обеспечивает обмен данными, например, с функциональным блоком начальной загрузки (BSF), опорным абонентским сервером (HSS) и сервером внешнего контента.

Процессор 410 может представлять собой, например, центральный процессор (CPU, central processing unit), микропроцессор, цифровой сигнальный процессор (DSP, digital signal processor), графический процессор и т.п. На фиг. 4 показан один процессор 410,

10 однако сервер 400 может содержать несколько процессоров.

Память 420 может, например, представлять собой энергонезависимую или энергозависимую память, такую как постоянное запоминающее устройство (ROM, read-only memory), программируемое постоянное запоминающее устройство (FROM, programmable read-only memory), стираемое программируемое постоянное запоминающее устройство (EPROM, erasable programmable read-only memory), оперативная память (RAM, random-access memory), флэш-память, диск для хранения данных, оптическое запоминающее устройство, магнитное запоминающее устройство, смарт-карта и т.п. Сервер 400 приложений может содержать несколько запоминающих устройств. Память 420 может быть выполнена в виде компонента сервера 400 приложений либо может

15 устанавливаться в слот, порт и т.п. сервера 400 приложений. Память 420 может использоваться только для хранения данных или может входить в состав устройства, служащего для других целей, например, для обработки данных.

Модуль 440 общей архитектуры начальной загрузки (GBA) может содержать функциональный блок сетевых приложений (NAF). GBA с целью аутентификации может

25 использоваться между функциональным блоком сетевых приложений (NAF) и UE, а также - между UE и функциональным блоком сетевых приложений (NAF) для обеспечения безопасности в тракте связи. По окончании начальной загрузки устройство UE и функциональный блок сетевых приложений (NAF) могут выполнять некоторый специфический для приложений протокол, согласно которому аутентификация

30 сообщений основывается на сеансовых ключах, сгенерированных во время взаимной аутентификации между UE и функциональным блоком сервера начальной загрузки (BSF).

Специалисту в этой области техники должно быть понятно, что помимо элементов, показанных на фиг.4, сервер 400 приложений может содержать и другие элементы,

35 например дополнительные схемы, такие как схемы ввода/вывода (I/O, input/output), схемы памяти, специализированные интегральные схемы (ASIC, application-specific integrated circuits), схемы обработки для конкретных задач, например схемы кодирования/декодирования исходного сигнала, схемы кодирования/декодирования канала, схемы шифрования/дешифрования и т.п.

На фиг. 5 показан пример блок-схемы пользовательского устройства 500, в котором могут применяться различные варианты осуществления настоящего изобретения. Такое устройство может представлять собой пользовательское оборудование (UE), пользовательскую аппаратуру, например мобильный терминал, портативный компьютер, планшетный компьютер или другое устройство связи.

40

Общая структура пользовательского устройства 500 включает в свой состав модуль 550 интерфейса связи, процессор 510, связанный с модулем 550 интерфейса связи, и память 520, связанную с процессором 510. Пользовательское устройство также содержит программное обеспечение 530, хранимое в памяти 520 и предназначенное для загрузки

45

в процессор 510 с целью выполнения. Программное обеспечение 530 может состоять из одного или более программных модулей и может быть реализовано в виде компьютерного программного изделия. Пользовательское устройство 500 также содержит контроллер 560 пользовательского интерфейса, связанный с процессором

5 510.

Модуль 550 интерфейса связи реализует по меньшей мере часть процесса передачи пользовательских данных, обсуждавшегося в связи с различными вариантами осуществления настоящего изобретения. Модуль 550 интерфейса связи может, например, представлять собой модуль радиointерфейса, такой как модуль радиосвязи в соответствии с технологиями WLAN, Bluetooth, GSM/GPRS, CDMA, WCDMA или LTE. Модуль 550 интерфейса связи может интегрироваться в пользовательское устройство 500 или в адаптер, плату и т.п., которые могут устанавливаться в подходящем слоте или порте пользовательского устройства 500. Модуль 550 интерфейса связи может поддерживать одну или множество технологий радиointерфейса. На фиг. 5 показан 10 один модуль 550 интерфейса связи, однако пользовательское устройство 500 может содержать множество модулей 550 интерфейсов связи.

Процессор 510 может представлять собой, например, центральный процессор (CPU, central processing unit), микропроцессор, цифровой сигнальный процессор (DSP), графический процессор и т.п. На фиг. 5 показан один процессор 510, однако 20 пользовательское устройство 500 может содержать несколько процессоров.

Память 520 может, например, представлять собой энергонезависимую или энергозависимую память, такую как постоянное запоминающее устройство (ROM), программируемое постоянное запоминающее устройство (PROM), стираемое программируемое постоянное запоминающее устройство (EPROM), оперативная память 25 (RAM), флэш-память, диск для хранения данных, оптическое запоминающее устройство, магнитное запоминающее устройство, смарт-карта и т.п. Пользовательское устройство 500 может содержать несколько запоминающих устройств. Память 520 может быть выполнена в виде компонента устройства 500 либо может устанавливаться пользователем в слот, порт и т.п. пользовательского устройства 500. Память 520 может 30 использоваться только для хранения данных или может входить в состав устройства, служащего для других целей, например, для обработки данных.

Универсальная карта 540 с интегральной схемой (UICC) может входить в состав пользовательского устройства в качестве смарт-карты. Универсальная карта 540 с интегральной схемой (UICC) обеспечивает целостность и безопасность использования 35 определенных персональных данных. На универсальной карте 540 с интегральной схемой (UICC) может храниться ее уникальный серийный номер, уникальный международный идентификатор мобильного абонента (IMSI), информация об аутентификации в целях безопасности и о шифровании, временная информация, связанная с локальной сетью, список услуг, к которым пользователю предоставлен 40 доступ, и пароли (PIN - для обычного использования и PUK - для разблокировки). Универсальная карта 540 с интегральной схемой (UICC) также может содержать несколько приложений, позволяющих одной смарт-карте предоставлять доступ к различным сетям, а также хранить телефонную книгу и другие приложения. В системе может использоваться встроенный модуль обеспечения безопасности для хранения и 45 обработки ключей.

Контроллер 560 пользовательского интерфейса может содержать схемы для приема входной информации от владельца пользовательского устройства 500, например, через клавиатуру, графический интерфейс пользователя, отображаемый на дисплее

пользовательского устройства 500, схемы распознавания речи или вспомогательные устройства, такие как телефонная трубка, а также в состав контроллера могут входить схемы для вывода информации, предназначенной пользователю, например, через графический интерфейс пользователя или динамик.

5 Специалисту в этой области техники должно быть понятно, что помимо элементов, показанных на фиг. 5, пользовательское устройство 500 может содержать и другие элементы, например микрофоны, дисплеи, а также дополнительные схемы, такие как
10 схемы ввода/вывода (I/O), схемы памяти, специализированные интегральные схемы (ASIC), схемы обработки для конкретных задач, например схемы кодирования/декодирования исходного сигнала, схемы кодирования/декодирования канала, схемы шифрования/дешифрования и т.п. Кроме того, пользовательское устройство 500 может содержать сменную или перезаряжаемую батарею (не показанную на чертеже) для питания пользовательского устройства 500 в отсутствие внешнего источника питания.

На фиг. 6 показан алгоритм выполнения операций в пользовательском устройстве
15 в соответствии с вариантом осуществления настоящего изобретения. Выполнение способа начинается на шаге 600. На шаге 610 принимается внешний код, содержащий запрос специфического для сервера ключа начальной загрузки (Ks_NAF). На шаге 620 определяется идентификатор сервера (NAF-Id). Специфический для сервера ключ начальной загрузки (Ks_NAF) генерируется на основе идентификатора сервера (NAF-
20 Id) на шаге 630. На шаге 640 определяется маркер доступа. На шаге 650 генерируется специфический для внешнего кода ключ начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа. На шаге 660 используется специфический для внешнего кода ключ начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего
25 кода. Выполнение способа завершается на шаге 670.

На фиг. 7 показан алгоритм выполнения операций в сервере приложений в соответствии с вариантом осуществления настоящего изобретения. Выполнение способа начинается на шаге 700. На шаге 710 передается код сценария, содержащий запрос специфического для кода сценария ключа начальной загрузки (Ks_js_NAF). На шаге
30 720 определяется идентификатор сервера (NAF-Id). На шаге 730 генерируется специфический для сервера ключ начальной загрузки (Ks_NAF) с использованием идентификатора сервера (NAF-Id). На шаге 740 определяется маркер доступа. На шаге 740 генерируется специфический для кода сценария ключ начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и
35 маркера доступа. На шаге 760 используется специфический для кода сценария ключ начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности кода сценария. Выполнение способа завершается на шаге 770.

Выше были представлены различные варианты осуществления настоящего изобретения. Следует принимать во внимание, что в этом документе каждое из таких
40 слов, как "содержит", "включает" и "состоит" используется как допускающее включение других признаков и не подразумевает наличия исключительно указанных признаков.

Выше с помощью некоторых примеров конкретных вариантов осуществления настоящего изобретения было приведено полное и информативное описание наилучшего пути его осуществления, как представляется в настоящее время авторам этого
45 изобретения. Однако специалисту в этой области техники должно быть понятно, что настоящее изобретение не ограничено представленной выше подробной информацией о вариантах его осуществления, и оно может быть реализовано в других вариантах с использованием эквивалентных средств или в различных сочетаниях вариантов без

отхода от его сути.

Кроме того, некоторые раскрытые выше признаки осуществления настоящего изобретения могут использоваться без соответствующего использования других признаков. Таким образом, приведенное описание должно рассматриваться просто в качестве иллюстрации принципов настоящего изобретения и не должно его ограничивать. Следовательно, объем изобретения ограничен только прилагаемой формулой изобретения.

Формула изобретения

1. Способ реализации механизма обеспечения безопасности для внешнего кода, включающий:

прием внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF);

определение идентификатора сервера (NAF-Id) и генерацию специфического для сервера ключа начальной загрузки (Ks_NAF) на основе идентификатора сервера (NAF-Id);

определение маркера доступа;

генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа и

использование специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

2. Способ по п. 1, включающий определение маркера доступа с использованием первого произвольного запроса (RAND1) и второго произвольного запроса (RAND2).

3. Способ по п. 2, включающий также:

передачу второго произвольного запроса (RAND2) и специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) в сервер приложений для проверки достоверности специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF).

4. Способ по п. 3, отличающийся тем, что шаг передачи включает также:

передачу ответного внешнего кода, содержащего второй произвольный запрос (RAND2) и специфический для внешнего кода ключ начальной загрузки (Ks_js_NAF).

5. Способ по п. 1, включающий:

прием внешнего кода приложением браузера устройства из сервера приложений;

определение идентификатора сервера (NAF-Id) и маркера доступа прикладным программным интерфейсом (JS-GBA-API) браузера приложения;

запрос специфического для сервера ключа начальной загрузки (Ks_NAF) прикладным программным интерфейсом (JS-GBA-API) из модуля начальной загрузки операционной системы;

прием специфического для сервера ключа начальной загрузки (Ks_NAF) прикладным программным интерфейсом (JS-GBA-API) из модуля начальной загрузки и

генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) прикладным программным интерфейсом (JS-GBA-API).

6. Способ по любому из пп. 1-5, включающий также:

установление туннеля обеспечения безопасности на транспортном уровне (TLS) между приложением браузера устройства и сервером приложений и

определение идентификатора сервера (NAF-Id), содержащего доменное имя (FQDN) и идентификатор протокола обеспечения безопасности.

7. Способ по п. 6, отличающийся тем, что идентификатор протокола обеспечения безопасности формируют с использованием набора шифров обеспечения безопасности на транспортном уровне (TLS).

8. Способ по любому из пп. 1-5, включающий также:

5 генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с помощью функции формирования ключа.

9. Способ по любому из пп. 1-5, отличающийся тем, что внешний код содержит код JavaScript.

10. Способ по п. 1, включающий также определение маркера доступа с использованием главного ключа обеспечения безопасности на транспортном уровне (TLS).

11. Устройство для реализации механизма обеспечения безопасности для внешнего кода, содержащее:

по меньшей мере один процессор и

15 по меньшей мере один модуль памяти, в котором хранится компьютерный программный код, при этом по меньшей мере один модуль памяти и компьютерный программный код сконфигурированы таким образом, чтобы при взаимодействии по меньшей мере с одним процессором устройство выполняло по меньшей мере следующие операции:

20 прием внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF);

определение идентификатора сервера (NAF-Id) и генерацию специфического для сервера ключа начальной загрузки (Ks_NAF) на основе идентификатора сервера (NAF-Id);

25 определение маркера доступа;

генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа и

30 использование специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

12. Устройство по п. 11, отличающееся тем, что по меньшей мере один модуль памяти и компьютерный программный код сконфигурированы таким образом, чтобы при взаимодействии по меньшей мере с одним процессором устройство выполняло:

35 определение маркера доступа с использованием первого произвольного запроса (RAND1) и второго произвольного запроса (RAND2).

13. Устройство по п. 11 или 12, отличающееся тем, что по меньшей мере один модуль памяти и компьютерный программный код сконфигурированы таким образом, чтобы при взаимодействии по меньшей мере с одним процессором устройство выполняло следующие операции:

40 прием внешнего кода приложением браузера устройства от сервера приложений;

определение идентификатора сервера (NAF-Id) прикладным программным интерфейсом (JS-GBA-API) браузера;

45 запрос специфического для сервера ключа начальной загрузки (Ks_NAF) прикладным программным интерфейсом (JS-GBA-API) из модуля начальной загрузки операционной системы;

прием специфического для сервера ключа начальной загрузки (Ks_NAF) прикладным программным интерфейсом (JS-GBA-API) из модуля начальной загрузки и

генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF)

прикладным программным интерфейсом (JS-GBA-API).

14. Устройство по п. 11 или 12, отличающееся тем, что по меньшей мере один модуль памяти и компьютерный программный код сконфигурированы таким образом, чтобы при взаимодействии по меньшей мере с одним процессором устройство выполняло:

5 определение маркера доступа с использованием главного ключа обеспечения безопасности на транспортном уровне (TLS).

15. Машиночитаемый носитель информации, содержащий компьютерную программу, содержащую компьютерный программный код, при исполнении которого по меньшей мере одним процессором устройство выполняет следующие операции:

10 прием внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF);

определение идентификатора сервера (NAF-Id) и генерацию специфического для сервера ключа начальной загрузки (Ks_NAF) на основе идентификатора сервера (NAF-Id);

15 определение маркера доступа;

генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа и

использование специфического для внешнего кода ключа начальной загрузки

20 (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

16. Способ реализации механизма обеспечения безопасности для внешнего кода, включающий:

передачу внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF);

25 определение маркера доступа;

генерацию специфического для сервера ключа начальной загрузки (Ks_NAF) с использованием идентификатора сервера (NAF-Id);

генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и

30 маркера доступа и

использование специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

17. Способ по п. 16, включающий:

запрос специфического для сервера ключа начальной загрузки (Ks_NAF) из

35 функционального блока сервера начальной загрузки (BSF); и

определение идентификатора сервера (NAF-Id), содержащего доменное имя (FQDN) и идентификатор протокола обеспечения безопасности.

18. Способ по п. 16 или 17, включающий:

прием специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF);

40 проверку достоверности специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) путем сравнения сгенерированного специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с принятым специфическим для внешнего кода ключом начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

45 19. Сервер приложений, содержащий:

по меньшей мере один процессор и

по меньшей мере один модуль памяти, в котором хранится компьютерный программный код, при этом по меньшей мере один модуль памяти и компьютерный

программный код сконфигурированы таким образом, чтобы при взаимодействии по меньшей мере с одним процессором сервер приложений выполнял по меньшей мере следующие операции:

передачу внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF);

генерацию специфического для сервера ключа начальной загрузки (Ks_NAF) с использованием идентификатора сервера (NAF-Id);

определение маркера доступа;

генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа и

использование специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

20. Машиночитаемый носитель информации, содержащий компьютерную программу, содержащую компьютерный программный код, при исполнении которого по меньшей мере одним процессором сервер приложений выполняет следующие операции:

передачу внешнего кода, содержащего запрос специфического для сервера ключа начальной загрузки (Ks_NAF);

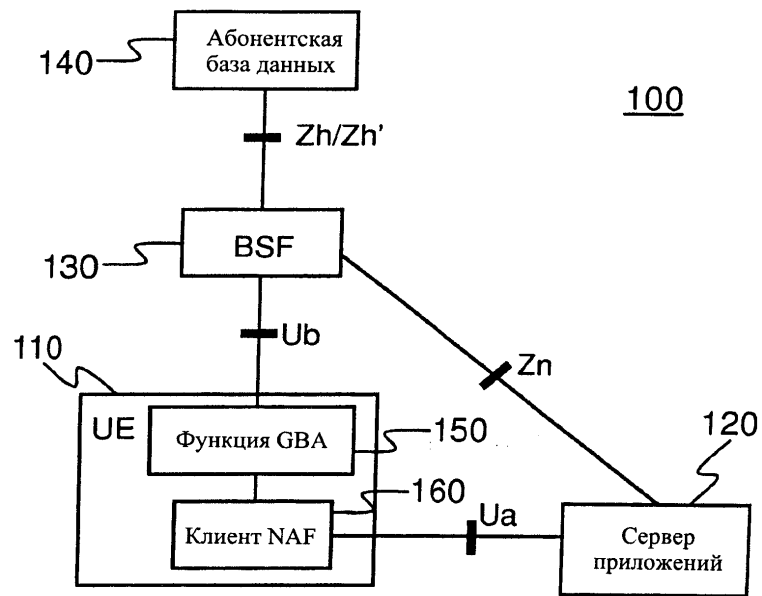
генерацию специфического для сервера ключа начальной загрузки (Ks_NAF) с использованием идентификатора сервера (NAF-Id);

определение маркера доступа;

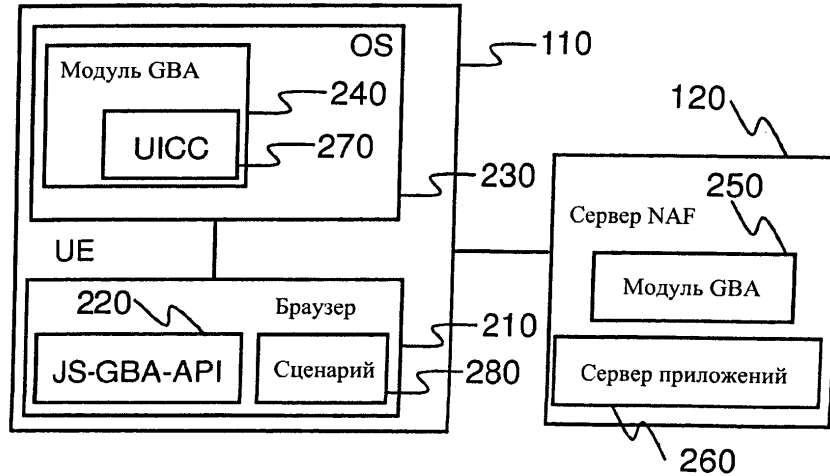
генерацию специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) с использованием специфического для сервера ключа начальной загрузки (Ks_NAF) и маркера доступа и

использование специфического для внешнего кода ключа начальной загрузки (Ks_js_NAF) для механизма обеспечения безопасности внешнего кода.

1 / 5

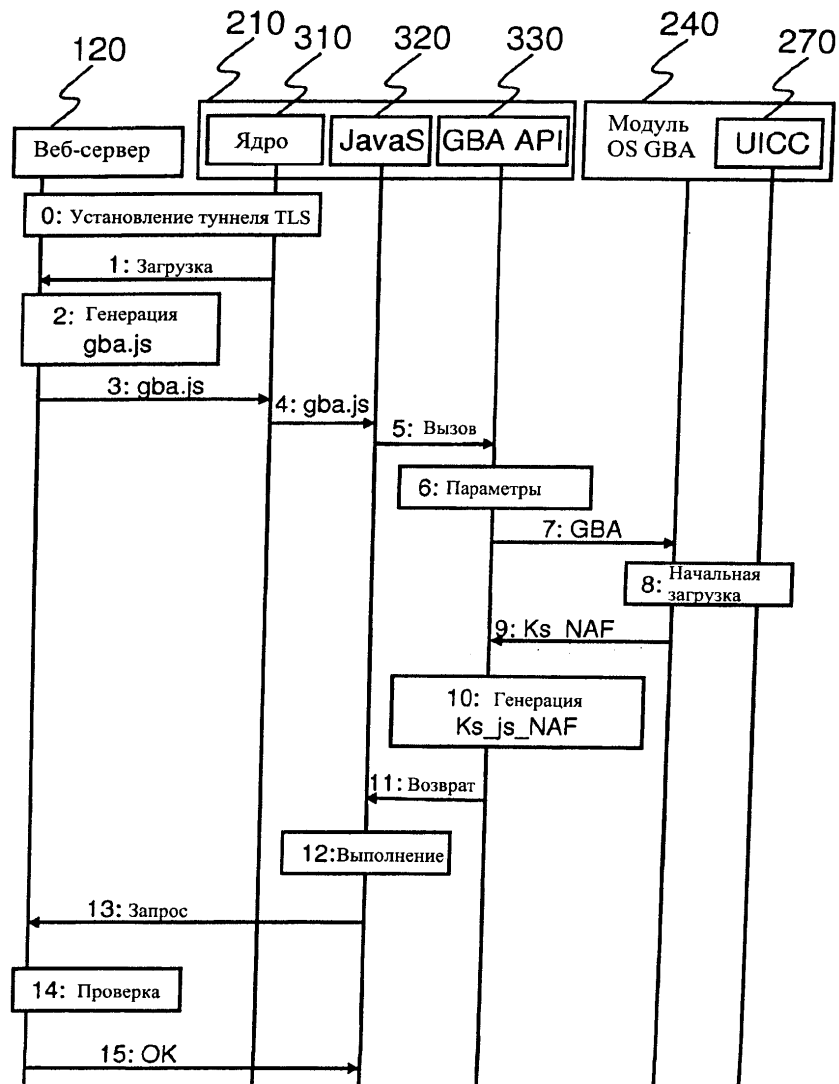


Фиг. 1



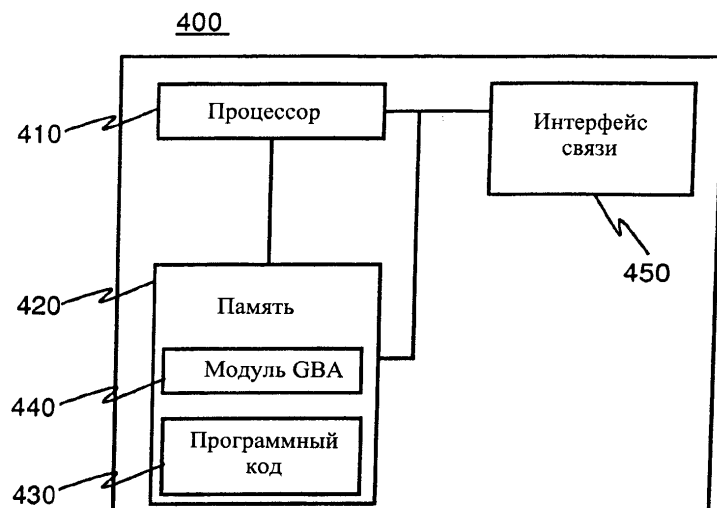
Фиг. 2

2 / 5

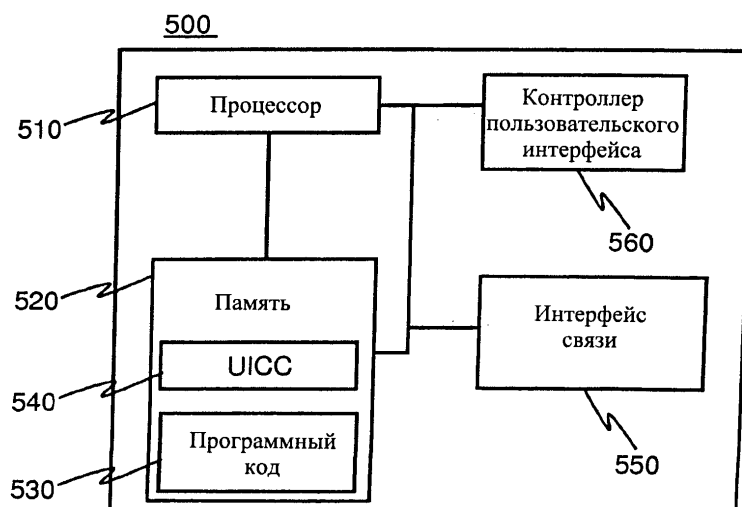


Фиг. 3

3 / 5

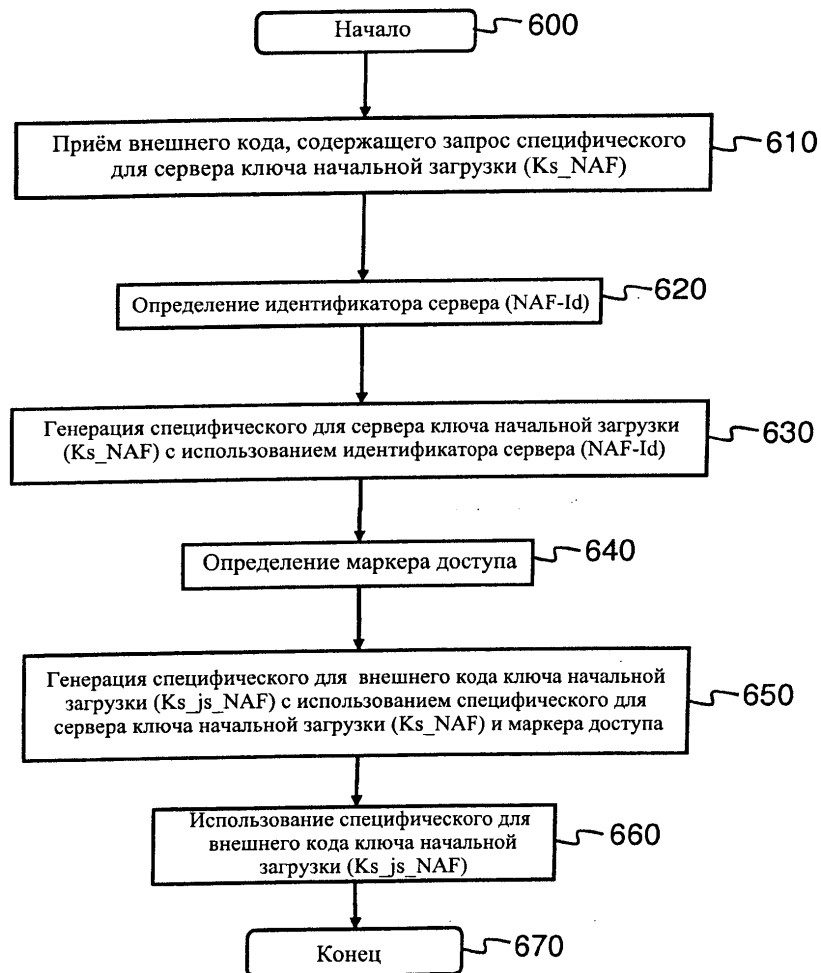


Фиг. 4



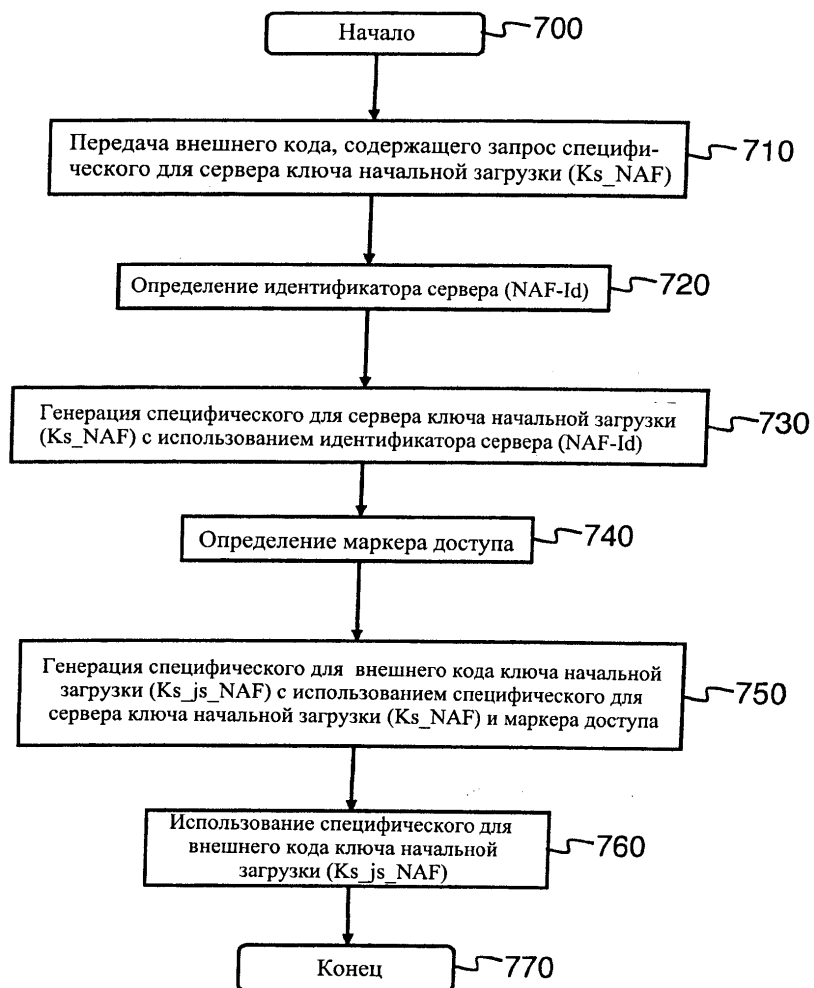
Фиг. 5

4 / 5



Фиг. 6

5 / 5



Фиг. 7