Протокол обмена персональными данными: ИКС

В. С. Бельский, И. Ю. Герасимов, К. Д. Царегородцев, И. В. Чижов

Аннотация—**Передача персональных данных и предотвращение их раскрытия являются широко распространенными проблемами в цифровом мире. Существует множество информационных технологий, внедренных в государственные и коммерческие сервисы. Людям необходимо передавать персональную информацию для использования этих технологий. И поэтому существенно важно обеспечить безопасность этой передачи. Несмотря на многочисленные правовые регулирования, существует множество случаев утечек персональных данных, которые привели к нежелательным последствиям. Причиной утечки может быть как атака злоумышленника, так и административная ошибка. В то же время, создание сложного взаимодействия с сервисом и многослойной системы информационной безопасности могут привести к неудобствам для пользователя. Протокол обмена персональными данными должен решать следующие задачи: передача персональных данных между участниками, обеспечение информационной безопасности, доверия к участникам сети и доступности сервисов. В работе представлен протокол обмена персональными данными: ИКС. Основная идея заключается в шифровании персональных данных на стороне пользователя и таким образом предотвращение раскрытия и публикации данных. Такой подход позволяет нам передавать персональные данные от пользователя сервису только лишь в форме зашифрованного пакета данных — блоба. Каждый блоб может быть проверен и заверен инспектором персональных данных который подтвердил информацию о пользователе. Инспектором может быть любой государственный департамент или коммерческая организация, например, центр выдачи паспорта, банки и др. Таким образом, мы можем обеспечить несколько ключевых особенностей для обмена персональными данными. Сервис, запрашивающий персональные данные, не может их опубликовать. Он все еще может выполнить протокол проверки с инспектором для подтверждения персональных данных. Мы не используем внутреннюю инфраструктуру сервиса и не усложняем работу инспектора за счёт добавления дополнительной информации о запросе персональных данных. Пакет персональных данных связывает личность владельца персональных данных и запрос сервиса. Каждый блоб создается для одного запроса и имеет временное ограничение для зашифрованных персональных данных. По истечении времени сервис не может использовать полученный набор данных. Пользователь не может предоставить неверные персональные данные или использовать данные другого пользователя. Мы не ограничиваем использование определенных криптографических алгоритмов. Протокол ИКС может быть реализован с любым алгоритмом шифрования, цифровой подписи и генерации ключа, которые стойки в нашей модели противника. Для описания протокола используются Российские криптографические стандарты. В статье также есть описание нескольких примеров того, как протокол ИКС может быть использован в существующих информационных системах.**

Ключевые слова—**ИКС, персональные данные, ВКО ГОСТ, симметричная криптография**

I. Введение

В последнее время информационные технологии активно внедряются в процессы оказания государственных услуг. Электронное правительство, порталы государственных услуг и подобные информационные системы становятся всё более привычными в современном обществе. Взять кредит, застраховать имущество или жизнь, оформить заграничный паспорт, заключить контракт с работодателем можно не выходя из дома, используя компьютер или смартфон. Как правило, при совершении этих операций гражданам необходимо предоставить свои персональные данные. Несмотря на достаточно строгое регулирование правил по обработке, хранению и передаче персональных данных во многих странах, постоянно происходят случаи их утечки в результате ошибок администратора или хакерских атак, что может приводить

Статья получена 19 мая 2020.

Бельский Владимир Сергеевич, Лаборатория Криптографии АО НПК «Криптонит», (email: v.belsky@kryptonite.ru).

Герасимов Илья Юрьевич, Лаборатория Криптографии АО НПК «Криптонит», МГУ им. М.В. Ломоносова (email: i.gerasimov@kryptonite.ru).

Царегородцев Кирилл Денисович, Лаборатория Криптографии АО НПК «Криптонит», (email: k.tsaregorodtsev@kryptonite.ru).

Чижов Иван Владимирович, Лаборатория Криптографии АО НПК «Криптонит», МГУ имени М.В. Ломоносова, Федеральный исследовательский центр «Информатика и управление» РАН (email: ichizhov@cs.msu.ru).

к нежелательным последствиям для людей, например, к потере денег, собственности и репутации.

Усложнение правил взаимодействия с сервисами путём увеличения количества и разнообразия средств информационной безопасности приводит к созданию существенных неудобств для пользователей таких сервисов, что, в первую очередь, снижает их привлекательность для граждан. Поэтому создание удобной и безопасной системы обработки персональных данных — одна из главных задач разработчиков государственных информационных сервисов.

В самом общем случае перед системами обработки персональных данных ставятся следующие задачи:

- Передача персональных данных между участниками;
- Обеспечение информационной безопасности;
- Обеспечение доверия к участникам системы;
- Обеспечение доступности сервисов.

Существует множество практик и сервисов, решающее поставленные задачи или связанное с ними. Одним из примеров является сетевой протокол аутентификации Kerberos [1]. Однако для его работы необходимо создание защищенного сервера выдачи тикетов. Компрометация сервера приведет к раскрытию всех данных пользователей, что приводит к единой точке отказа и дополнительным требованиям к инфраструктуре. Другим примером является протокол авторизации OAuth [2], однако его особенностью является передача прав сервису по паролю, что накладывает ответственность на сервис за обеспечение передачи, обработки и хранения персональных данных. Если же пользователь передал персональные данные сервису по защищенному каналу, необходим высокий уровень доверия этому сервису, что не может быть гарантированно, когда сервис шифрует на своей стороне персональные данные пользователя, а сам пользователь теряет контроль над обработкой переданной личной информации.

Для решения упомянутых выше задач в статье описывается протокол ИКС безопасной передачи персональных данных пользователей различным сервисам. В статье приводится описание протокола, в том числе алгоритмов регистрации участников, механизмов передачи и проверки персональных данных, структур передаваемых данных, и даются рекомендации по криптографическим алгоритмам, которые могут быть использованы в протоколе.

II. О персональных данных

Проблема утечек персональных данных возникла давно, однако в последнее время она вышла на совершенно новый уровень. Использование облаков, электронных сервисов, систем с удаленной регистрацией стало не просто обычным, а необходимым занятием. Количество информации, собранной о пользователях различными сервисами и службами, принимает угрожающие размеры.

В эпоху цифрового общества отправка копии паспортных данных или номера СНИЛС по почте или через мессенджеры становится обычным делом. Например, регистрация в различных сервисах кратковременной аренды автомобилей требует отправления фотокопий паспорта и прав пользователя. Многие IT-компании собирают и накапливают значительное количество информации о пользователях, в итоге утечки этих данных приобретают масштабы локальных катастроф.

Среди значимых утечек можно выделить случаи описанные в статьях [3, 4]. По всему миру за последнее время количество записей персональных данных, замешанных в различных утечках, превысило миллиард [5]. Одновременно растет число предложений о продажах баз данных с персональными сведениями [6].

Законодательно эту проблему решают принятием определенных норм, устанавливающих требования на тех, кто персональные данные собирает и обрабатывает. Например, в России принят закон о персональных данных [7], который регулирует указанную область и устанавливает требования и ограничения. В Европе принят закон под названием GDPR [8], который содержит достаточно строгие положения о контроле обработки персональных данных и устанавливает крупные штрафы за несоблюдение этих требований. Правоприменительная практика только складывается, тем не менее уже существуют прецеденты, в которых при нарушении законодательства были выписаны значительные штрафы нарушителям [9][10]. При этом очевидно, что применение одних законодательных норм не может полностью решить проблему защиты персональных данных. И в поддержку юридических норм необходимо применять технические средства защиты.

Технически проблему защиты персональных данных решить непросто. Так, например, только шифрование данных не может решить проблему, поскольку сервис, который использует персональные данные, получает к ним доступ, а значит он может потенциально нарушить их

конфиденциальность. Возникает необходимость обеспечить высокий уровень доверия к этому сервису. Но как это сделать техническими средствами? Кроме того, при передаче данных пользователь полностью теряет контроль над процессом их обработки. Известные случаи утечек персональных данных, как правило, возникают именно после передачи пользователем данных сервису, и в момент, когда сервис не смог обеспечить должный уровень безопасности.

Существуют протоколы, которые совместно с аутентификацией позволяют предоставлять сервису доступ к некоторым данным. Одним из примеров является сетевой протокол аутентификации Kerberos [1], который решает задачу аутентификации и предоставления доступа пользователям к определенным ресурсам. Протокол использует только симметричные алгоритмы шифрования и состоит из нескольких процедур аутентификации и получения «билетов» — специальных структур данных, которые дают права доступа. При этом задачу дальнейшего контроля обработки переданной информации протокол не решает.

Другим примером является протокол авторизации OAuth [2]. Его особенностью является передача прав сервису на основе парольной аутентификации. Этот протокол также не решает задачу контроля обработки переданной информацией. Заметим, что протокол OAuth используется для аутентификации сервисов с использованием учетной записи на «Портале государственных услуг Российской Федерации». Следует отметить, что в Российской Федерации существуют планы по созданию единого цифрового профиля гражданина на основе этого портала, где в одном месте будут собраны все основные персональные данные, а пользователь получит единый идентификатор. На первый взгляд, такой подход мог бы решить проблему утечек, если везде, где необходимо передавать персональные данные, вместо них будет передаваться только единый идентификатор. Однако такое решение проблемы несёт новые угрозы. Примером может служить существующая в США система, которая в качестве единого идентификатора использует номер социального страхования SSN. В случае утечки этого номера злоумышленник получает возможность распоряжаться почти всеми персональными данными, а значит, злоумышленник при этом фактически ворует цифровую личность человека. Более того, организация надёжной защиты единой базы персональных данных всех пользователей является очень сложной задачей, и указанный подход уже получил соответствующую критику российского регулятора в области защиты информации [11].

Одним из подходов к техническому решению вопроса является использование современных криптографических методов. Так, концепция гомоморфного шифрования или протоколов конфиденциальных вычислений предназначены для обеспечения возможности обработки данных без доступа к ним. При этом нужно учитывать, что подобные криптографические примитивы плохо изучены, обладают крайне низкими потребительскими свойствами, а также в настоящее время нет ни одного международного или национального стандарта на них. Использование же стандартизованных примитивов — обычное требование в государственных информационных системах.

В случае персональных данных сервису, как правило, не нужно их обрабатывать, сервису достаточно знать, что эти данные получены от пользователя, они актуальные и правильные. Предложенный в работе протокол ИКС предлагает решение задачи безопасного использования персональных данных пользователей стандартизованными средствами и протоколами.

III. Назначение протокола

Предположим, что персональные данные выдаются пользователю некоторым инспектором (иногда в литературе встречается термин «провайдер»), который всегда может эти данные проверить и подтвердить. Персональные данные выдаются сервису на время, по истечении которого они перестают быть валидными, т. е. теперь использовать их для совершения каких-либо операций (например, подписание договора) нельзя. Единственная возможность, которая остаётся у сервиса — в любой момент проверить персональные данные пользователя у инспектора и убедиться, что эти данные были верны и соответствовали конкретному пользователю в некоторый момент времени в прошлом.

Сформулируем ряд свойств, которые должен обеспечить протокол, решающий задачу безопасной передачи персональных данных от пользователя к сервису:

1) У сервиса должна быть возможность удостовериться, что персональные данные поступили от определенного пользователя;
2) У сервиса должна быть возможность проверить, что персональные данные валидны (корректны, актуальны, соответствуют запрашиваемому типу данных и соответствуют пользователю);
3) У сервиса нет возможности передать персональные данные третьим лицам;

4) Пользователь не может предоставить некорректные, неактуальные или чужие персональные данные;
5) Пользователь не может предоставить персональные данные, не соответствующие запросу сервиса;
6) Пользователь предоставляет персональные данные на ограниченный срок. По истечении срока сообщение с персональными данными становится невалидным;
7) Пользователь может проверить кому, на какое время, и какие именно персональные данные он предоставляет;
8) Пользователь может оспорить факт предоставления персональных данных, если он этого не делал;
9) Пользователь не может отказаться от факта предоставления персональных данных, если он их действительно предоставлял.

Указанные свойства протокола ИКС достигаются использованием криптографических механизмов. Персональные данные передаются сервису строго в шифрованном виде. Расшифровать их способен только пользователь, который и формирует шифрованный набор данных (блоб), или инспектор, который должен быть способен проверить эти данные. Для обеспечения аутентификации участников и проверки целостности сообщений используются механизмы электронной подписи. Таким образом, предполагается, что где-то развёрнута инфраструктура управления открытыми ключами пользователей, предполагающая наличие удостоверяющего центра, который регистрирует пользователей и обеспечивает соответствие открытых ключей их владельцам. Сама инфраструктура не является частью протокола, поэтому её описание выходит за рамки статьи.

IV. Участники протокола

В протоколе задействованы следующие участники:

U: Пользователь системы, владелец персональных данных. Пользователь должен быть зарегистрирован в удостоверяющем центре **CA**;

Inspector: Инспектор персональных данных. Доверенная служба, которая подтверждает сервису соответствие персональных данных, предъявляемых пользователем, идентификатору пользователя. В качестве инспектора может, например, выступать государственный орган (например, налоговая служба или пенсионный фонд) или коммерческая структура (например, банк или страховая компания).

Src: Сервис запрашивает у пользователя персональные данные. В качестве сервиса может, например, выступать медицинское учреждение, финансовая организация или любая коммерческая компания, с которой пользователь вступает в договорные отношения.

AgentU: Пользовательский агент — программа или устройство, посредством которой пользователь взаимодействует с участниками протокола. В качестве агента обычно выступает браузер или мобильное приложение.

CA: Удостоверяющий центр. Служба, выполняющая функции по обеспечению жизненного цикла сертификатов открытых ключей и управлению учётными данными участников протокола. Предполагается, что все участники регистрируются в удостоверяющем центре. Более подробно процесс регистрации будет описан ниже.

V. Используемые обозначения

В дальнейшем будем использовать следующие обозначения и договорённости:

$E_K(M)$**:** Зашифрование сообщения M на ключе K с помощью симметричного алгоритма шифрования;

K_A**:** Ключ формирования подписи (закрытый ключ) участника протокола A;

$Cert_A$**:** Ключ проверки подписи (открытый ключ) участника протокола A, представленный в виде сертификата, подписанного **CA**; будем считать, что сертификату ключа подписи $Cert_A$ соответствует ключ подписи (секретный ключ) K_A;

$Sign(M)_{Cert_A}$**:** Электронная подпись под сообщением M, полученная с помощью ключа формирования подписи участника A, соответствующего $Cert_A$;

$(a_1, \ldots, a_n, sign_A)$**:** Набор сообщений $a_1, \ldots, a_n$ и электронная подпись под этим набором $sign_A = Sign(a_1, \ldots, a_n)_{Cert_A}$, объединённые в вектор;

x_A**:** Закрытая часть эфемерного ключа абонента A, которая используется при выполнении процедуры согласования ключей;

y_A**:** Открытая часть эфемерного ключа абонента A, которая используется при выполнении процедуры согласования ключей, $y_A = x_A \cdot P$, P — ненулевая точка циклической подгруппы группы точек эллиптической кривой;

K_{AB}**:** Общий симметричный ключ шифрования между участниками протокола A и B.

VI. Описание протокола ИКС

Протокол состоит из следующих основных этапов:

1) Регистрация участников протокола
2) Передача персональных данных
3) Проверка персональных данных

Общая схема протокола указана на рисунке 1.

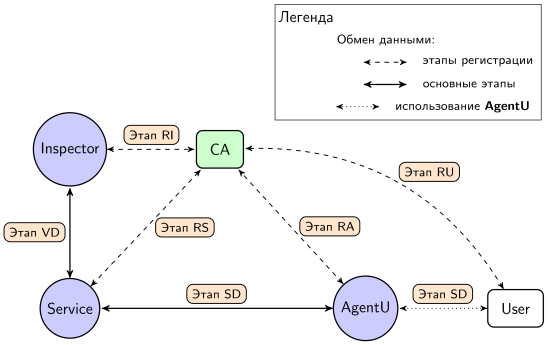

Рис. 1. Схема протокола

Предполагается, что взаимодействие между участниками осуществляется по общедоступной сети (Интернет), защищённой средствами протокола TLS [12] или другого аналогичного ему протокола. Кроме того, предполагается, что участники протокола имеют защищенный доступ к сервисам **CA**, необходимым для выполнения проверки подписи под сообщениями. Например, к таким сервисам относятся службы получения актуальных сертификатов открытых ключей, обновления отозванных сертификатов и т. п. Далее не будем отдельно останавливаться на этих действиях участников, упоминая лишь факт проверки электронной подписи с помощью заверенного сертификата открытого ключа.

Опишем шаги протокола, отвечающие за регистрацию участников. Сразу отметим, что этапы регистрации могут выполняться в любом порядке и независимо друг от друга. Исходный код нашей практической реализации протокола доступна в GitHub: https://github.com/RnD-Kryptonite/x-protocol. Репозиторий содержит возможную реализацию шагов протокола ИКС на языке Python с описанием каждого участника протокола в виде соответствующего класса.

A. Этап регистрации пользователя (Этап RU)

Для регистрации пользователь выполняет следующие шаги (см. рисунок 2):

1) Регистрируется в **CA**, подтверждая свою личность документом, удостоверяющим личность (необходимо физическое присутствие пользователя).
2) Устанавливает пароль к своему аккаунту в **CA**.
3) Получает идентификатор **UID**. В дальнейшем будем считать, что **UID** может выступать как логин пользователя в **CA**.

Рис. 2. Этап RU

B. Этап регистрации агента (Этап RA)

В качестве агента (**AgentU**) используется приложение на телефоне (планшете) или браузер на компьютере. Для регистрации **AgentU** необходимо выполнить следующие действия (см. рисунок 3):

1) Пользователь, используя приложение **AgentU**, аутентифицируется в **CA** и посылает запрос на регистрацию приложения.
2) В случае успешной аутентификации **CA** отправляет **AgentU** подтверждение аутентификации и запрашивает открытый ключ пользователя.
3) Приложение **AgentU** генерирует открытый и закрытый ключ подписи пользователя $Cert_U$ и K_U. Открытый ключ по защищенному каналу направляется в **CA**.
4) **CA** подтверждает корректность открытого ключа пользователя, подписывает сертификат открытого ключа и публикует его в списке сертификатов **CA**. В приложение **AgentU** направляется подписанный сертификат $Cert_U$.

C. Этап регистрации сервиса (Этап RS)

Для регистрации сервис выполняет следующие шаги (см. рисунок 4):

1) Направляет запрос на регистрацию в **CA**, указывая свои необходимые регистрационные данные и сетевое имя (адрес) для направления запросов к сервису от пользователей.
2) Получает в ответ зарегистрированный идентификатор **SrcID**.

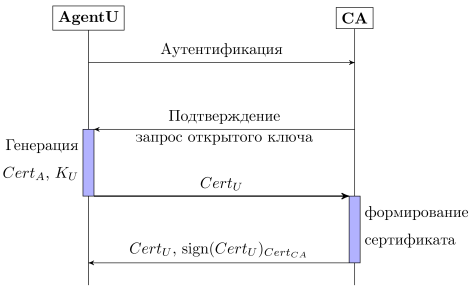

Рис. 3. Этап RA

3) Генерирует открытый и закрытый ключ подписи $Cert_S$ и K_S. Открытый ключ по защищенному каналу направляется в **CA**.
4) **CA** подтверждает корректность открытого ключа, подписывает сертификат открытого ключа и публикует его в списке сертификатов **CA**. В сертификат включается сетевое имя (адрес) для направления запросов к сервису. Сервису направляется подписанный сертификат $Cert_S$.

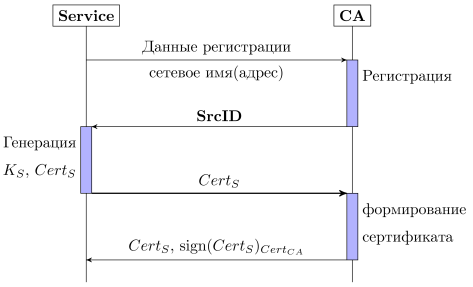

Рис. 4. Этап RS

D. Этап регистрации инспектора (Этап RI)

Для регистрации инспектор выполняет следующие шаги (см. рисунок 5):

1) Направляет запрос на регистрацию в **CA**, указывая необходимые регистрационные данные и сетевое имя (адрес) для направления запросов от сервиса. Кроме того, инспектор указывает **scope**, который он будет проверять.
2) Получает в ответ зарегистрированный идентификатор **InID**.
3) Генерирует закрытый и открытый ключ подписи K_I и $Cert_I$. Кроме того, инспектор генерирует закрытый и открытый ключ для процедуры согласования ключей x_I и y_I. Открытый ключи $Cert_I$ и y_I по защищенному каналу направляет в **CA**.
4) **CA** подтверждает корректность открытого ключа инспектора, подписывает сертификат открытого ключа и публикует его в списке сертификатов **CA**. В сертификат включается сетевое имя (адрес) для направления запросов к инспектору и y_I. Инспектору направляется подписанный сертификат $Cert_I$.

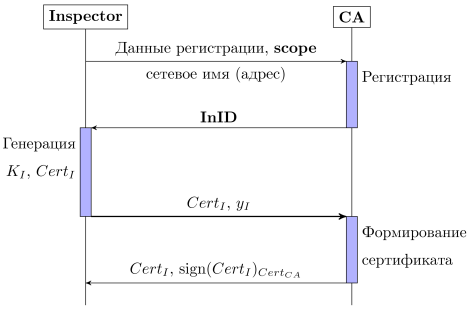

Рис. 5. Этап RI

При описании протокола предполагается, что сервис и инспектор представляют собой некоторые организации, которые обладают необходимыми информационными системами и техническими средствами для взаимодействия с **CA** и реализации криптографических механизмов. Конкретный порядок взаимодействия с **CA** и необходимые для этого средства зависят от реализации самого **CA**, их описание выходит за рамки этой статьи. Для протокола важно, что все участники зарегистрированы и имеют корректные пары открытых и закрытых ключей подписи.

Пользователь не всегда обладает средствами для обеспечения корректной работы криптографических механизмов, именно поэтому для пользователя определен **AgentU** в качестве способа взаимодействия с другими участниками и реализации необходимых криптографических преобразований.

Кроме этого, предполагается, что **CA** по **scope** публикует инспектора, которые имеют право проверять конкретный набор персональных данных. Например, это можно реализовать, включив идентификатор набора персональных данных в сертификат открытого ключа инспектора. Корректность и непротиворечивость соответствия **scope** инспектору является ответственностью **CA**. В целом, протокол не фиксирует механизм обеспечения такого соответствия.

E. Этап передачи персональных данных (Этап SD)

Опишем как происходит передача персональных данных от пользователя к сервису. Пользователь **U** обращается в **Src** за некоторой услугой (выписать справку, оформить кредит, заключить договор). Обращение происходит через **AgentU**, например, на сайте сервиса. Для получения услуги пользователю, как правило, необходимо передать свои персональные данные сервису, причем сервис должен быть уверен, что полученные им данные принадлежат именно пользователю и впоследствии (в случае необходимости или при соблюдении некоторых условий) могут быть раскрыты.

Для передачи персональных данных пользователь и сервис выполняют следующие действия (см. рисунок 6):

1) Пользователь через **AgentU** отправляет **UID** сервису **Src** по адресу, указанному в $Cert_S$.
2) Сервис **Src** формирует запрос на получение персональных данных **REQUEST**, в котором указывает **SrcID**, **UID**, **scope**, **TTL**, и подписывает его с помощью закрытого ключа K_S. Запрос отправляется обратно в приложение **AgentU**.

$$\textbf{REQUEST} = \\ \big(\textbf{SrcID}, \textbf{UID}, \textbf{scope}, \textbf{TTL}, sign_S\big)$$

TTL представляет собой параметр, содержащий текущую метку времени и информацию о том на какое время сервису необходим доступ к данным.
3) По полученному запросу **AgentU** проверяет валидность подписи сервиса под **REQUEST**.
4) Пользователь через приложение проверяет корректность **SrcID**, **scope** и **TTL**. Другими словами, пользователь визуально контролирует кому, на какое время и какие данные он собирается выдать.
5) **AgentU** определяет инспектора, который будет проверять запрошенные данные, получает из $Cert_I$ открытый ключ инспектора y_I для выполнения согласования ключей. **AgentU** выполняет процедуру VKO_GOST и получает K_{UI} — общий симметричный ключ шифрования канала связи с инспектором. Для этого приложение пользователя **AgentU** генерирует эфемерные открытый ключ y_U и закрытый ключ x_U и вычисляет $K_{UI} = K(x_U, y_I, UKM)$. Описание функции $K(\cdot,\cdot,\cdot)$ приведено в таблице I.
6) **AgentU** формирует ответ **REPLY**, который состоит из полученного ранее запроса **REQUEST**, значений персональных данных **SecData** и случайного числа (соли) $salt$, которое вырабатывает **AgentU** специально для запроса. При этом ответ шифруется на ключе K_{UI}:

$$\textbf{REPLY} = \\ E_{K_{UI}}(\textbf{REQUEST}, \textbf{SecData}, salt).$$

7) **AgentU** формирует блоб (**blob**) и подписывает его с помощью закрытого ключа подписи K_U:

$$\textbf{blob} = (y_U, \textbf{UID}, \textbf{REPLY}, sign_U)$$

AgentU отправляет **blob** сервису.

На этом этап передачи персональных данных сервису закончен. В итоге сервис получает **blob**, под которым он может проверить подпись с помощью опубликованного $Cert_U$ и удостовериться, что блоб сформирован именно пользователем **U**. При этом расшифровать закрытую часть **REPLY** сервис не может, поскольку не имеет ключа K_{UI}. У пользователя остался подписанный сервисом запрос **REQUEST**.

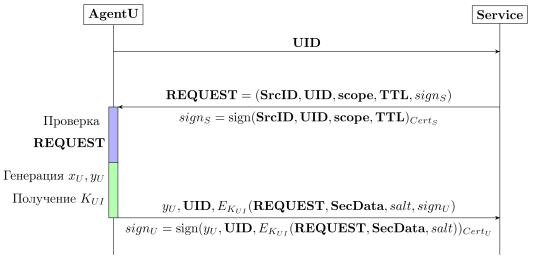

Рис. 6. Этап SD

Таблица I
Согласование ключа по алгоритму VKO

VKO_GOSTR3410	
m — порядок группы точек эллиптической кривой, q — порядок циклической подгруппы, P — ненулевая точка подгруппы, UKM — случайное число.	
A	B
$x_A, y_A = x_A \cdot P$	$x_B, y_B = x_B \cdot P$
$K(x_A, y_B, UKM) =$ $\left(\frac{m}{q} \cdot UKM \cdot x_A \bmod q\right) \cdot (y_B)$	

F. Этап проверки персональных данных (Этап VD)

Проверка персональных данных может быть выполнена сервисом в любой момент, после

того как он получил **blob**, подписанный пользователем. Для проверки персональных данных сервис выполняет следующие действия (см. рисунок 7):

1) Сервис определяет инспектора по значению **scope**, которое есть в его исходном запросе, и отправляет **blob** $= (y_U, \mathbf{UID}, \mathbf{REPLY}, sign_U)$ для проверки инспектору **Inspector**. Сетевой адрес инспектора сервис получает из сертификата $Cert_I$.
2) **Inspector** проверяет подпись пользователя под полученным **blob** с помощью опубликованного $Cert_U$
3) **Inspector** извлекает y_U из блоба и, используя закрытую часть своего ключа x_I, выполняет процедуру VKO_GOST для получения общего с пользователем **U** симметричного ключа K_{UI}, вычисляя $K_{UI} = K(x_I, y_U, UKM)$. Описание функции $K(\cdot,\cdot,\cdot)$ приведено в Таблице I.
4) **Inspector** расшифровывает **REPLY** $= E_{K_{UI}}(\mathbf{REQUEST}, \mathbf{SecData}, salt)$ на ключе K_{UI} и получает **REQUEST**, **SecData** и $salt$.
5) **Inspector** проверяет, что идентификатор пользователя, полученный из сертификата $Cert_U$ на шаге 2, совпадает с идентификатором пользователя из **REQUEST**. В противном случае завершает протокол ошибкой.
6) **Inspector** проверяет, что время, тип персональных данных и сервис, который обратился за проверкой, соответствуют значением **TTL**, **scope** и **SrcID** из запроса **REQUEST**. Если что-то из этого неверно, завершает протокол ошибкой.
7) По значения **SecData**, **TTL** и **UID** инспектор проверяет корректность указанных персональных данных на момент времени **TTL** с помощью внутренней базы данных инспектора. Принимается решение о валидности персональных данных: yes/no.
8) По результатам проверки инспектор направляет сервису следующий набор данных:
   $(\mathbf{blob}, \mathbf{TTL}, yes/no, sign_I)$.

Необходимость использования при проверке **TTL** связана с тем, что персональные данные могли измениться (например, смена фамилии).

VII. Анализ свойств безопасности протокола

В разделе III приведены свойства протокола, которые должны выполняться. Проведём анализ протокола и выделим механизмы, которые обеспечивают выполнение требуемых свойств безопасности.

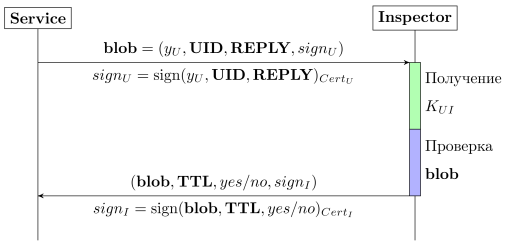

Рис. 7. Этап VD

Свойство 1 (у сервиса должна быть возможность удостовериться, что персональные данные поступили от определенного пользователя) обеспечивается проверкой цифровой подписи пользователя под сформированным блобом, а также цифровой подписью инспектора под результатом проверки блоба. При этом связь цифровой подписи пользователя с конкретным человеком обеспечивается на этапе физической регистрации пользователя в **CA**.

Свойство 2 (у сервиса должна быть возможность проверить, что персональные данные валидны) обеспечивается проверкой **blob** у инспектора. Заметим, что результат проверки блоба заверен цифровой подписью инспектора, принадлежность которой к конкретному инспектору подтверждает **CA**.

Свойство 3 (у сервиса нет возможности разгласить персональные данные) выполняется, поскольку сервис получает персональные данные в зашифрованном виде. Ключ шифрования блоба создан с помощью процедуры VKO_GOST между пользователем **U** и инспектором **InID** и неизвестен сервису.

Выполнение свойства 4 (пользователь не может предоставить некорректные персональные данные) обеспечивается проверкой инспектором соответствия **UID**, **TTL** и **SecData** на этапе проверки персональных данных.

Рассмотрим свойство 5 (пользователь не может предоставить персональные данные, не соответствующие запросу сервиса) более подробно. Нарушение его выполнения заключается в несоответствии **SecData** полученному **REQUEST**. Если пользователь предоставляет некорректные персональные данные, то свойство 5 выполняется аналогично свойству 4. Рассмотрим атаку, при которой пользователь подменяет **REQUEST**. Успешная замена позволила бы пользователю оспорить факт предоставления блоба.

Пусть пользователь получил два **REQUEST**: R_a, R_b и хочет сформировать **blob** по R_a, но

указать в блобе запрос R_b. Рассмотрим следующие случаи:

- R_b : $\mathbf{scope}_{R_b}! = \mathbf{scope}_{R_a}$. Тогда **blob** не пройдет шаг 6 этапа проверки данных.
- R_b : $\mathbf{TTL}_{R_b}! = \mathbf{TTL}_{R_a}$. Рассмотрим две ситуации:
  - $\mathbf{SecData}_{R_b} == \mathbf{SecData}_{R_a}$, то есть персональные данные пользователя не менялись в течение промежутка времени $|\mathbf{TTL}_{R_b} - \mathbf{TTL}_{R_a}|$ между двумя запросами. В таком случае пользователь не нарушает работу протокола, повторно формируя аналогичный блоб на тот же запрос, но в другой промежуток времени.
  - $\mathbf{SecData}_{R_b}! = \mathbf{SecData}_{R_a}$. Тогда **blob** не пройдет шаг 7 этапа проверки данных.
- R_b : $\mathbf{SrcID}_{R_b}! = \mathbf{SrcID}_{R_a}$. В таком случае два разных сервиса запросили одинаковый **scope** персональных данных. Формирование блоба будет идентично под оба запроса, но при этом **blob** не пройдет шаг 6 этапа проверки данных.

Таким образом, пользователь не может реализовать угрозу при попытке подмены **REQUEST** в рамках работы протокола, и свойство 5 выполняется.

Свойство 6 (пользователь предоставляет персональные данные на ограниченный срок) выполняется, поскольку по истечении срока, указанного в блобе (параметр **TTL**), блоб с персональными данными не проходит шаг 6 этапа проверки данных и становится невалидным.

Выполнение свойства 7 (пользователь может проверить кому, на какое время, и какие именно персональные данные он предоставляет) обеспечивается процедурой формирования блоба. Заметим, что в запрос **REQUEST** включаются поля **SrcID**, **scope**, **TTL**. Этот запрос подписан с помощью закрытого ключа подписи, привязанного к сервису **Src**, поэтому пользователь уверен кому он передает данные. Кроме того, в блоб включены параметры **scope** и **TTL**, которые указывают какие данные и на какое время передаются сервису.

Свойство 8 (пользователь может оспорить факт предоставления персональных данных, если он этого не делал) обеспечивается следующим. Правильность блоба **blob** подтверждается инспектором только при выполнении ряда условий (шаги 2, 6 и 7 этапа проверки данных), в частности, успешной проверки корректности подписи пользователя под сформированным blob. Если пользователь не предоставлял данных, то подпись под блобом не может быть сформирована от его имени злоумышленником.

Свойство 9 (пользователь не может отказаться от факта предоставления персональных данных, если он их действительно предоставлял) выполняется согласно похожим рассуждениям. Наличие цифровой подписи пользователя под блобом, в котором содержатся запрашиваемые персональные данные, обеспечивает невозможность отказа пользователя от факта формирования блоба. А значит, согласно свойству 7, пользователь знал кому, на какое время и какие данные он предоставляет, и предоставил их.

VIII. Особенности протокола

Рассмотрим отдельные особенности протокола, которые могут быть использованы при построении соответствующей инфраструктуры обмена персональными данными.

Прежде всего заметим, что компрометация идентификатора пользователя не ведет к раскрытию всех персональных данных пользователя. Под компрометацией следует понимать опубликование информации о соответствии **UID** конкретному человеком. В протоколе идентификатор **UID** является частью сертификата $Cert_U$ и может быть доступен публично, однако в открытом виде связка **UID** и персональных данных не появляется. Более того, не существует единого места, где хранятся все персональные данные пользователя в связке с этим идентификатором. При этом, если скомпрометирован **blob** и эфемерный ключ x_U, на котором пользователь зашифровал персональные данные, противник получит доступ только к данным, указанным в блобе.

Заметим также, что пользователь самостоятельно шифрует свои персональные данные, и только он отвечает за этот процесс. Тем самым, безопасность этих данных не зависит от правил и порядков обращения с персональными данными у сервиса. При этом считается, что инспектор является доверенной стороной для остальных участников протокола.

Далее отметим, что для участия в протоколе инспектору необходимо реализовать лишь механизм проверки зашифрованного набора персональных данных, и нет необходимости серьезно менять свою инфраструктуру. Конечно, определенная нагрузка на инспектора присутствует, но это не идет в сравнение со сложностью организации, защиты и поддержки единых баз персональных данных пользователей.

Протокол не накладывает ограничения на криптографические алгоритмы и позволяет использовать любые протоколы цифровой подписи, алгоритмы шифрования и создания ключевого материала. В том числе, в протоколе ИКС

могут быть использованы следующие российские стандартизованные криптографические механизмы:

- ВКО Р 50.1.113-2016 (согласование ключей) [13];
- ГОСТ 34.10-2018 (электронная цифровая подпись) [14];
- ГОСТ 34.12-2018 (блочные шифры) [15].

IX. Возможные применения протокола

На шагах 6, 7 этапа передачи персональных данных пользователь выполняет шифрование своих персональных данных на общем с инспектором ключе K_{UI}. Формат персональных данных может быть любым, соответствующий запрошенному типу данных **scope**. Таким образом, протокол ИКС имеет гибкие возможности по его использованию в различных сценариях. В этом разделе приводятся лишь некоторые из них.

Все сценарии отличаются множеством **scope** запрашиваемых персональных данных, которое может состоять из:

- данных, необходимых для выполнения операций в интересах пользователя;
- идентификационных данных пользователя (ФИО, логин, идентификатор в некоторой системе).

A. Заключение трудовых договоров и совершение выплат зарплаты

Пользователь заключает трудовой договор с работодателем. Для этого он обязан предоставить большое количество персональных данных от имени до ИНН (идентификационный номер налогоплательщика). Кроме того, пользователь должен написать заявление о перечислении зарплаты на его счёт в банке.

Опишем, как могла бы выглядеть процедура заключения договора с использованием протокола ИКС. Работодатель играет роль сервиса. Он запрашивает персональные данные пользователя для заключения договора и формирования заявления о номере зарплатного счёта. В качестве инспектора в части персональных данных могут выступать государственные службы, а в части данных расчётного счёта — банк. Сервис указывает в запросе необходимые ему персональные данные пользователя. Пользователь, выполняя протокол ИКС, формирует два блоба, первый — с данными для заключения трудового договора, а другой — с данными расчётного счёта, в котором указывается распоряжение о перечислении любых денег, поступающих от сервиса, на такой-то расчётный счёт. Сервис выполняет протокол с инспектором (или несколькими инспекторами) и получает подтверждение блобов. Теперь в трудовой договор включается результат проверки первого блоба.

Для перечисления зарплаты сервис отправляет в банк, который выступал инспектором для проверки второго блоба, указание о перечислении зарплаты и второй блоб. Банк раскрывает блоб и выполняет указание из него перечислить сумму из заявки сервиса на счёт пользователя.

B. Покупка в интернет-магазине

Пользователь совершает покупку в интернет-магазине. Для оформления доставки и создания счёта ему необходимо предоставить паспортные данные, телефон и адрес доставки. Однако пользователь не доверяет интернет-магазину. Есть опасения, что персональные данные могут быть разглашены или переданы третьим лицам с целью навязывания разного рода рекламы или услуг.

Опишем, как могла бы выглядеть процедура покупки при использовании протокола ИКС. Интернет-магазин является сервисом, который запрашивает персональные данные пользователя для оформления счёта и доставки товара. Имеется инспектор — почта. Она может подтвердить соответствие паспортных данных и почтового адреса пользователю. Сервис указывает в запросе необходимые ему персональные данные пользователя. Если пользователь соглашается на оформление покупки, он следует протоколу и передает сервису блоб с этими данными. Сервис формирует посылку и отправляет её почте, прикладывая блоб, полученный от пользователя. Инспектор выполняет этап проверки персональных данных (этап VD), фактически раскрывая персональные данные пользователя и его адрес, и отправляет посылку по адресу пользователя.

C. Оформление кредита

Пользователь хочет взять кредит в банке. Банку необходимо проверить множество фактов о пользователе, его кредитную историю, наличие судимости и т. п. Для этого банку необходимо подтверждение пользователя и его персональные данные, например, номер паспорта и дату его выдачи, адрес регистрации, ИНН, СНИЛС и т. п. Однако пользователь может не доверять банку и опасается утечки своих данных.

Опишем вариант решения этой проблемы, если бы использовался протокол ИКС. Пользователь передает зашифрованные персональные

данные в виде блоба **blob** банку, который выступает в роли сервиса. Банк отсылает **blob** инспектору персональных данных (например, налоговой службе, МВД, пенсионному фонду и т. д.), который проверяет **blob**, и при успешной проверке предоставляет банку подтверждение персональных данных пользователя. В договоре о кредите между банком и пользователем указывается результат проверки.

Теперь банк имеет возможность обращаться к инспектору с просьбой предоставить какую-либо информацию о пользователе, данные которого указаны в блобе **blob**. Инспектор, раскрывая **blob**, может получить дополнительную информацию о нём из своей базы данных и потом предоставить её банку. После просрочки блоба банк уже не сможет получить никакой информации о пользователе.

D. Обращение в страховую компанию

Пользователь хочет получить услугу страхования жизни и здоровья. Страховой компании необходимо подтверждение состояния здоровья пользователя. Однако пользователь не доверяет компании и опасается раскрытия информации об его истории болезней.

Протокол ИКС может быть применен здесь следующим образом. Страховая компания, выполняющая роль сервиса, запрашивает у пользователя пустой блоб, то есть от пользователя не требуются какие-либо персональные данные, а только подтверждение. Другими словами, страховая запрашивает у пользователя справку, что он «здоров». Если пользователь согласен предоставить эту справку, он отправляет пустой блоб, сформированный и подписанный согласно протоколу. Этот блоб передается инспектору, которым является поликлиника. Получив блоб, инспектор проверяет историю болезней пользователя. Таким образом, результат проверки блоба определяется состоянием пользователя. Если пользователь проходит такую проверку (например, может управлять транспортным средством), то блоб успешно подтверждается на этапе VD. В договоре о страховании может указываться результат проверки, подтверждающий успешное выполнение протокола.

Отличительной особенностью этого примера является формирование пустого блоба, когда **SecData** является пустым множеством.

X. Направления дальнейших исследований

Предложенная версия протокола не решает задачу анонимности пользователей и сервисов. Любой инспектор получает доступ к информации какой пользователь какому сервису передал свои персональные данные. Частично эту задачу может решить анонимизация сервисов, т. е. инспектор не будет знать какому конкретно сервису предоставлены данные, а будет знать только его идентификатор. Это, однако, требует разработки протокола, использующего временные идентификаторы пользователей и сервисов, уникальные для каждого блоба.

Другим направлением исследования может стать использование личностной криптографии (IBC) для замены сертификатов открытых ключей в рамках протокола на идентификаторы IBC. Указанное изменение упростит работу с сертификатами.

На шаге 7 этапа проверки данных инспектор проверяет соответствие персональных данных **SecData** и идентификатора пользователя **UID**. При этом указанный идентификатор получен пользователем при регистрации в **CA** и не обязательно должен совпадать с идентификатором пользователя в информационной системе инспектора. Тогда, во-первых, если инспектор — государственный орган, то идентификатор пользователя **UID** может совпадать с тем идентификатором, который принят для учета пользователей в государственных базах данных (СНИЛС, SSN и т. п.). Такое соответствие может быть обеспечено на шаге регистрации пользователя при взаимодействии **CA** и инспектора. Во-вторых, может быть использован дополнительный шаг протокола, при котором **CA** сообщит инспектору дополнительную информацию для идентификации **UID** во внутренней базе инспектора. В-третьих, если это поддерживается инспектором, значение **SecData** в ответе может быть подписано пользователем с идентификатором, который представлен в формате, «понятном» для информационной системы инспектора. Выбор конкретного решения во многом зависит от практической реализации протокола и требует дальнейших исследований.

Как замечено выше, в случае компрометации закрытого ключа инспектора сервисы получат доступ к содержимому всех блобов инспектора. В протоколе используется одна пара ключей каждого инспектора для всех пользователей и всех блобов. Для уменьшения последствий в случае компрометации этого ключа можно использовать отдельные ключи инспектора для каждого пользователя. В этом случае инспектор должен организовать и поддерживать инфраструктуру управления ключами пользователя и опубликовать сервис, обеспечивающий доступ пользователей к открытому ключу инспектора.

Решения задачи диверсификации ключа также требует отдельных исследований.

XI. Стойкость протокола

A. Введение

В данном разделе мы докажем, что при некоторых предположениях выполнены сформулированные в разделе III свойства. А именно, мы хотим показать, что выполнены следующие условия:

1) Неразличимость ответов (reply indistinguishability):

a) Требование: противник не должен получать информацию о персональных данных пользователя, изучая его ответы $Reply$:

$$Reply = E_{K_{UI}}(Request, SecData, salt),$$

Мы не требуем, чтобы была скрыта длина персональных данных (количество шифруемых блоков), ID проверяющего инспектора.

b) Смысл требования: Сервис не должен иметь возможность разгласить персональные данные и вообще получить какую-либо информацию о персональных данных, изучая ответы Пользователя на свои запросы.

c) Что случится, если требование нарушено: Противник теоретически получит возможность извлекать информацию из ответов на свои запросы или запросы сервиса, что может привести к тому, что противник получит доступ к персональным данным пользователя.

2) Неподделываемость блоба (blob unforgeability):

a) Требование: противник не должен уметь подделывать блоб пользователя

$$[y, UID, Reply]_{sgn(U)}$$

b) Смысл требования:

1) У сервиса должна быть возможность удостовериться, что персональные данные поступили от определенного пользователя U;
2) Пользователь может оспорить факт предоставления персональных данных, если он этого не делал;
3) Пользователь не может отказаться от факта предоставления персональных данных, если он их действительно предоставлял.

c) Что случится, если требование нарушено: Противник теоретически получит возможность подделывать ответы пользователя на запросы сервиса, в частности, предоставлять за него персональные данные нежелательным сервисам.

3) Неподделываемость запроса (request unforgeability):

a) Требование: противник не должен уметь подделывать запрос Сервиса

$$Request = [SrcID, UID, scope, ttl]_{sgn(S)}$$

b) Смысл требования: Пользователь может проверить кому, на какое время, и какие именно персональные данные он предоставляет;

c) Что случится, если требование нарушено: Противник теоретически получит возможность подделывать запросы сервиса, в частности, запрашивать от имени сервиса персональные данные некоторых пользователей. Это может привести к нежелательным последствиям.

4) Неподделываемость ответа инспектора:

a) Требование: противник не должен уметь подделывать ответ инспектора

$$check = [yes/no, ttl, blob]_{sgn(I)}$$

b) Смысл требования:

1) У сервиса должна быть возможность проверить, что персональные данные валидны (корректны, актуальны, соответствуют запрашиваемому типу данных и соответствуют пользователю);
2) Пользователь не может предоставить некорректные, неактуальные или чужие персональные данные;
3) Пользователь не может предоставить персональные данные, не соответствующие запросу сервиса;
4) Пользователь предоставляет персональные данные на ограниченный срок. По истечении срока сообщение с персональными данными становится невалидным;

c) Что случится, если требование нарушено: Противник теоретически получит возможность подделывать ответы инспектора, в частности, подтверждать невалидные/просроченные персональные данные.

B. Основной результат

Указанные выше свойства выполняются в следующей модели противника:

1) Режим работы блочного шифра, используемого при обработке сообщений, является **ROR-CPA**-стойким;
2) Схема подписи, используемая в протоколе, является **EUF-CMA**-стойкой (в модели случайного оракула, см. [16]);
3) Задача Диффи-Хеллмана в соотстветствующей группе (в случае стандарта [13] — группе точек эллиптической кривой) является труднорешаемой;

А именно, верна следующая теорема:

Теорема 1. *Вероятность взлома протокола может быть оценена сверху как:*

$$\begin{gathered} 2q \cdot Adv^{DDH}(t + q \cdot (4 + 2\mu)) + \\ + q \cdot Adv_{SE}^{ROR-CPA}(t + q \cdot (2\mu + 1), 1, \mu) + \\ + 3 \cdot Adv_{Sert}^{\textbf{EUF-CMA}}(q_{usr}, t) + Adv_{insp}^{\textbf{EUF-CMA}}(q_{insp}, t) + \\ + Adv_{User}^{\textbf{EUF-CMA}}(q_{blob}, t) + Adv_{src}^{\textbf{EUF-CMA}}(q_{src}, t) \end{gathered} \quad (1)$$

где использованы обозначения:

q_{usr} — *количество пользователей, заверенных в* ***CA****;*

q_{blob} — *количество блобов, подписанных пользователем;*

q_{src} — *количество запросов, подписанных сервисом;*

q_{insp} — *количество запросов, подписанных инспектором;*

q — *количество запросов* Req *противника на зашифрование;*

μ — *длина наибольшего запрошенного* $Request \| SecData \| salt$;

t — *время, затраченное противником;*

C. Основные обозначения

Введем (и напомним) некоторые обозначения, которые будут использоваться в дальнейшем.

Для сокращения записи будем обозначать за $[M]_{sgn(A)}$ — подпись сообщения M с помощью закрытого ключа пользователя A;

Напомним также, что ранее были введены следующие обозначения:

$Request$**:** Запрос Сервиса вида

$$Request = [SrcID, UID, scope, ttl]_{sgn(S)}$$

$Reply$**:** Ответ Пользователя вида

$$Reply = E_{K_{UI}}(Request, SecData, salt)$$

$blob$**:** Сформированный блоб вида

$$blob = [y, UID, Reply]_{sgn(U)}$$

Введем также некоторые дополнительные обозначения, которые понадобятся в дальнейшем:

$check$**:** Ответ инспектора на запрос проверки блоба

$$check = [blob, ttl, yes/no]_{sgn(I)}$$

$GetData(UID, scope)$**:** Функция, которая возвращает персональные данные пользователя с идентификатором UID, соответствующие $scope$.

$GetID(scope)$**:** Функция, которая возвращает ID инспектора по $scope$.

Algorithm 1 Rnd

function INIT
  $b \leftarrow_R \{1, \ldots, |G|\}$
function $\mathcal{O}$
  $a \leftarrow_R \{1, \ldots, |G|\}$
  $c \leftarrow_R \{1, \ldots, |G|\}$
  return (g^a, g^b, g^c)

Algorithm 2 DH

function INIT
  $b \leftarrow_R \{1, \ldots, |G|\}$
function $\mathcal{O}$
  $a \leftarrow_R \{1, \ldots, |G|\}$
  return (g^a, g^b, g^{ab})

D. Основные задачи

Приведем список стандартных предположений, на которых будет основываться доказательство стойкости протокола. Всюду далее предполагается, что противник — это некоторый вероятностый алгоритм (вероятностная машина Тьюринга).

1) Распознавательная задача Диффи-Хеллмана (Decisional Diffie-Hellman, DDH): Пусть противнику дана циклическая группа G и порождающий ее элемент g. Противнику требуется отличить случайный элемент группы g^c от элемента, полученного в результате работы протокола Диффи-Хеллмана g^{ab} при известных g^a, g^b. Формально, противник взаимодействует с одним из экспериментаторов DH или $Rand$. В начале эксперимента в процедуре $Init$ фиксируется некоторое b (моделирующее секретный ключ второго участника). Затем противник делает один запрос к оракулу $\mathcal{O}$ и получает в зависимости от эксперимента либо тройку (g^a, g^b, g^{ab}), либо тройку (g^a, g^b, g^c). В задаче DDH противнику разрешается сделать лишь 1 запрос к своему оракулу $\mathcal{O}$. По выданной тройке противник должен определить, с каким из экспериментаторов он взаимодействует. Противник выдает бит 1, если он думает, что взаимодействует с оракулом DH, иначе он должен выдать бит 0.

Преимущество противника определяется как разность следующих вероятностей:

$$Adv^{DDH}(\mathcal{A}) = \mathbb{P}[DH(\mathcal{A}) \to 1] - \mathbb{P}[Rnd(\mathcal{A}) \to 1]$$

Вероятность берется по всем случайным вы-

борам в ходе эксперимента и по случайным битам самого противника $\mathcal{A}$.

$$Adv^{DDH}(t) = \max_{\mathcal{A} \in A(t)} Adv^{DDH}(\mathcal{A}),$$

максимум берется по всем противникам $\mathcal{A} \in A(t)$, которые работают время не более t. Еще раз отметим, что количество запросов к оракулу $\mathcal{O}$ равно 1.

2) Кратная распознавательная задача Диффи-Хеллмана (Multiple Decisional Diffie-Hellman, MDDH): Противнику, как и ранее, дана циклическая группа G и порождающий ее элемент g. В отличие от предыдущего эксперимента, противнику предоставляется возможность несколько раз обращаться к оракулу $\mathcal{O}$, то есть противник получает q троек вида:

$$(g^{x_1}, g^y, g^{z_1}), \ldots, (g^{x_q}, g^y, g^{z_q}),$$

где z_i либо равен $x_i \cdot y$ (во всех тройках), либо выбирается случайно равновероятно (во всех тройках). Заметим, что в экспериментах DH и Rnd элемент $b(= y)$ выбирается 1 раз.

Задача противника состоит в том, чтобы различить эти две ситуации:

$$Adv^{MDDH}(\mathcal{A}) = \mathbb{P}[DH(\mathcal{A}) \to 1] - \mathbb{P}[Rnd(\mathcal{A}) \to 1]$$

$$Adv^{MDDH}(t, q) = \max_{\mathcal{A} \in A(t,q)} Adv^{MDDH}(\mathcal{A}),$$

максимум берется по всем противникам $\mathcal{A} \in A(t, q)$, которые работают время не более t и делают не более q запросов к оракулу $\mathcal{O}$.

Утверждение 1. *Выполнено следующее неравенство:*

$$Adv^{MDDH}(t, q) \leq q \cdot Adv^{DDH}(t + 4q)$$

Доказательство. С помощью гибридного аргумента (см., например, [17], [16]) покажем, что неравенство выполнено.

Будем обозначать $\xi_i = (g^{x_i}, g^y, g^{x_i \cdot y})$ — случайные величины, полученные от оракула $\mathcal{O}$ в эксперименте DH; $\eta_i = (g^{x_i}, g^y, g^{z_i})$ — случайные величины, полученные от оракула $\mathcal{O}$ в эксперименте Rnd.

Тогда необходимо найти расстояние между случайными векторами (различить два случайных вектора):

$$\xi = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_q \end{bmatrix}, \eta = \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_q \end{bmatrix},$$

где расстояние понимается в смысле неразличимости двух случайных векторов алгоритмом, работающем время не более t :

$$\Delta(\xi, \eta) = \max_{\mathcal{A}} |\mathbb{P}[\mathcal{A}(\xi) \to 1] - \mathbb{P}[\mathcal{A}(\eta) \to 1]|.$$

Заметим, что для таким образом введенного расстояния выполняется неравенство треугольника:

$$\Delta(\xi, \eta) \leq \Delta(\xi, \nu) + \Delta(\nu, \eta),$$

поскольку для каждого конкретного фиксированного алгоритма $\mathcal{A}$ выполняется:

$$\begin{aligned} &|\mathbb{P}[\mathcal{A}(\xi) \to 1] - \mathbb{P}[\mathcal{A}(\eta) \to 1]| = \\ &= |\mathbb{P}[\mathcal{A}(\xi) \to 1] - \mathbb{P}[\mathcal{A}(\nu) \to 1] + \\ &+ \mathbb{P}[\mathcal{A}(\nu) \to 1] - \mathbb{P}[\mathcal{A}(\eta) \to 1]| \leq \\ &\leq |\mathbb{P}[\mathcal{A}(\xi) \to 1] - \mathbb{P}[\mathcal{A}(\nu) \to 1]| + \\ &+ |\mathbb{P}[\mathcal{A}(\nu) \to 1] - \mathbb{P}[\mathcal{A}(\eta) \to 1]| \end{aligned} \tag{2}$$

Переходя сначала в правой части неравенства к максимуму по всем допустимым $\mathcal{A}$, а затем в левой части, приходим к требуемому неравенству.

Рассмотрим следующий набор векторов:

$$\gamma_i = \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_i \\ \xi_{i+1} \\ \vdots \\ \xi_q \end{bmatrix}.$$

Тогда $\gamma_0 = \xi, \ldots, \gamma_q = \eta$.

В таком случае, мы можем расписать:

$$\Delta(\gamma_0, \gamma_n) \leq \Delta(\gamma_0, \gamma_1) + \ldots + \Delta(\gamma_{q-1}, \gamma_q),$$

где данное неравенство выполнено в силу неравенства треугольника (см. выше).

Следовательно, если мы ограничим вероятность различения двух «соседних» векторов γ_i и γ_{i+1}, то вероятность различения γ_0 и γ_q будет не превосходить суммы вероятностей различения «соседних» векторов.

Покажем, что $\Delta(\gamma_i, \gamma_{i+1})$ можно ограничить сверху через сложность задачи DDH.

При переходе $\gamma_i \to \gamma_{i+1}$ мы заменяем координату ξ_{i+1} на η_{i+1}, оставляя остальные координаты неизменными. Если найдется противник $\mathcal{B}$, который хорошо различает замену одной из координат, то его можно будет использовать для решения задачи DDH. Для этого достаточно промоделировать все остальные координаты вектора, а $(i + 1)$-ю координату получить от оракула DDH.

Для моделирования предположим, что у нас есть оракул $\mathcal{O}$ (из задачи DDH), который выдает либо тройку (g^x, g^y, g^{xy}), либо (g^x, g^y, g^z). Построим по указанному выше противнику $\mathcal{B}$ противника $\mathcal{A}$, который решает задачу DDH.

Сгенерируем первые i координат вектора:

1) Сгенерируем

$$\begin{gathered} x_t \leftarrow_R \{1, \ldots, |G|\}, \\ z_t \leftarrow_R \{1, \ldots, |G|\}, t = 1, \ldots, i; \end{gathered} \tag{3}$$

2) Положим $\eta_t = (g^{x_t}, g^y, g^{z_t}), t = 1, \ldots, i;$

В качестве $(i+1)$-й координаты поставим ответ оракула — тройку (g^x, g^y, C), где $C = g^{xy}$ либо $C = g^r$ выбрано случайно (в зависимости от того, с каким оракулом на самом деле мы взаимодействуем).

Оставшиеся координаты сгенерируем следующим образом:

1) Сгенерируем

$$\begin{gathered} x_t \leftarrow_R \{1, \ldots, |G|\}, \\ t = (i+2), \ldots, q; \end{gathered} \tag{4}$$

2) Положим

$$\begin{gathered} \xi_t = (g^{x_t}, g^y, (g^y)^{x_t}), \\ t = (i+2), \ldots, q; \end{gathered} \tag{5}$$

В таком случае координаты векторов будут иметь необходимые распределения. На моделирование требуется сгенерировать не более $2q$ случайных чисел и возвести элементы группы в соответствующие степени (не более $2q$ элементов).

Таким образом, на моделирование требуется не более чем $4q$ операций (генерация случайного числа, возведение в степень в группе). Если считать, что указанные выше операции производятся за 1 шаг работы вычислительного устройства, то получим оценку:

$$\Delta(\gamma_i, \gamma_{i+1}) \leq Adv^{DDH}(t + 4q),$$

и следовательно, итоговая оценка:

$$Adv^{MDDH}(t, q) = \Delta(\gamma_0, \gamma_q) \leq q \cdot Adv^{DDH}(t+4q)$$

□

*3) Модель **ROR-CPA**:* Дадим формальное определение модели **ROR-CPA** для схемы шифрования SE с вектором инициализации IV, который выбирается случайным при каждом зашифровании.

Противник взаимодействует с одним из экспериментаторов: $Real$ или $Rand$. Экспериментатор $Real$ выбирает секретный ключ (на шаге инициализациии) и предоставляет доступ к оракулу зашифрования. Экспериментатор $Rand$ также выбирает секретный ключ, но его оракул вместо зашифрования поданного ему сообщения выбирает случайную строку той же длины и зашифровывает её.

Algorithm 3 Rand

```
function Init
    k ←_R Keys
function O(m)
    iv ←_R IV
    r ←_R {0,1}^|m|
    return SE_k^iv(r)
```

Algorithm 4 Real

```
function Init
    k ←_R Keys
function O(m)
    iv ←_R IV
    return SE_k^iv(m)
```

Если противник думает, что он взаимодействует с оракулом настоящего шифрования, то он должен выдать 1, иначе выдать 0. Преимущество противника в модели **ROR-CPA** определяется как разность соответствующих вероятностей:

$$\begin{gathered} Adv_{SE}^{ROR-CPA}(\mathcal{A}) = \\ = \mathbb{P}[Real(\mathcal{A}) \to 1] - \mathbb{P}[Rand(A) \to 1] \end{gathered} \tag{6}$$

*4) **EUF-CMA**-модель схемы подписи:* Дадим формальное определение **EUF-CMA**-модели схемы подписи.

Экспериментатор $Forge$ в ходе инициализации выбирает пару секретный и открытый ключ и предоставляет доступ противнику к оракулу подписи.

На каждый запрос m от противника $\mathcal{A}$ оракул запоминает запрошенное сообщение и отдает подпись к нему. Противник делает некоторое количество запросов к оракулу $\mathcal{O}$, после чего выдает сообщение m^* и подпись σ^*.

Экспериментатор возвращает 1 в том и только том случае, когда подпись верна и сообщение m^* не было запрошено ранее.

Algorithm 5 Forge

function INIT
  $(pk, sk) \leftarrow_R Keys$
  $msgs \leftarrow [\,]$
function $\mathcal{O}(m)$
  $msgs \leftarrow msgs \cup m$
  $\sigma = Sign(sk, m)$
  return (m, σ)
function FIN$((m^*, \sigma^*))$
  if $m \in msgs$ **then**
    return 0
  return $Verify(pk, m^*, \sigma^*)$

Противник имеет доступ к оракулу подписи $\mathcal{O}$ и может адаптивно подбирать сообщения для подписи $m_1, \ldots, m_q$. Цель противника — получить подпись под сообщением, которое не было запрошено ранее. Преимущество противника в модели **EUF-CMA** определяется как вероятность подделки подписи:

$$Adv^{\textbf{EUF-CMA}}(q, t) = \max_{\mathcal{A} \in A(q,t)} \mathbb{P}[Forge(\mathcal{A}) \to 1]$$

5) Дальнейшее доказательство: Используя введенные выше модели $MDDH$, **ROR-CPA** и **EUF-CMA**, ограничим сверху вероятность нежелательных событий через преимущество противника в указанных моделях. Для этого рассмотрим каждое из требований (неразличимость ответов, неподделываемость блоба, неподделываемость запроса, неподделываемость ответа инспектора), формализуем его в виде соответствующего эксперимента и ограничим сверху преимущество противника в каждом из экспериментов через преимущество противника в базовых моделях ($MDDH$, **ROR-CPA** и **EUF-CMA**).

E. Неразличимость ответов (reply indistinguishability)

a) Требование: противник не должен получать информацию о персональных данных пользователя, изучая его ответы $Reply$:

$$Reply = E_{K_{UI}}(Request, SecData, salt),$$

Мы не требуем, чтобы была скрыта длина персональных данных (количество шифруемых блоков), ID проверяющего инспектора.

b) Чем обеспечивается:

1) **ROR-CPA** стойкость режима шифрования $E_K(\cdot)$;
2) **DDH**-задача для соответствующей группы, используемой в протоколе согласования ключей ([13]);

c) Формализация требования: Противник имеет право подавать оракулу $\mathcal{O}$ пары запросы вида:

$$Request = (SrcID, UID, scope, ttl)$$

Оракул по $scope$ определяет Инспектора для проверки данных и проводит с ним протокол выработки общего секретного ключа $k = K_{UI}$. Процедура выработки общего секретного ключа происходит при каждом новом запросе.

На запрос оракул $Real$ отвечает:

$$Reply = E_k(Request, SecData, salt)$$

На запрос оракул $Rand$ отвечает:

$$Reply = E_k(Request, r, salt), \;\; |r| = |SecData|$$

Оракул возвращает результат противнику. Изучая ответы оракула, противник должен выдать бит 0, если он считает, что взаимодействует с оракулом $Rand$, иначе выдать бит 1.

Algorithm 6 Rand

function $\mathcal{O}(Request)$
  $m \leftarrow GetData(UID, scope)$
  $InspID \leftarrow GetID(scope)$
  $k \leftarrow VKO(K_U^{priv}, K_I^{publ})$
  $r \leftarrow_R \{0,1\}^{|m|+saltLen}$
  return $E_k(Request \| r)$

Algorithm 7 Real

function $\mathcal{O}(Request)$
  $m \leftarrow GetData(UID, scope)$
  $InspID \leftarrow GetID(scope)$
  $k \leftarrow VKO(K_U^{priv}, K_I^{publ})$
  $salt \leftarrow \{0,1\}^{saltLen}$
  return $E_k(Request \| m \| salt)$

Определяем преимущество противника как разность соответствующих вероятностей:

$$\begin{gathered} Adv^{ROR-Reply}(\mathcal{A}) = \\ = \mathbb{P}[Real(\mathcal{A}) \to 1] - \mathbb{P}[Rand(\mathcal{A}) \to 1] \end{gathered} \tag{7}$$

$$\begin{gathered} Adv^{ROR-Reply}(t, q, \mu) = \\ \max_{\mathcal{A} \in A(t,q,\mu)} Adv^{ROR-Reply}(\mathcal{A}), \end{gathered} \tag{8}$$

где максимум берется по всем противникам $\mathcal{A} \in A(t, q, \mu)$, которые:

1) Работают время не более t;
2) Делают не более q запросов к оракулу $\mathcal{O}$;
3) Длина наибольшего запрошенного $Request \| SecData \| salt$ не более μ.

Мы предполагаем, что функция $GetData$ по UID и $scope$ возвращает персональные данные пользователя, привязанного к UID, соответствующие $scope$. Функция $GetID$ по запросу $scope$ возвращает ID инспектора, проверяющего персональные данные, соответствующие запросу $scope$.

Утверждение 2. *Преимущество противника в задаче различения ответов может быть ограничено сверху как:*

$$Adv^{ROR-Reply}(t,q,\mu) \le 2 \cdot MDDH(t+2q\mu,q)+ \\ +q \cdot Adv_{SE}^{ROR-CPA}(t+q\cdot(2\mu+1),1,\mu) \quad (9)$$

Доказательство. Доказательство проведем в два этапа.

1) На первом этапе заменим согласование ключей по протоколу VKO на случайно выбираемый ключ. При этом оценкой сверху будет сложность задачи **MDDH** для соответствующей группы (например, для группы точек эллиптической кривой из стандарта [13]).
2) На втором этапе сведем различение $Reply$ со случайными ключами к **ROR-CPA**-стойкости шифра $E_k(\cdot)$.

Algorithm 8 Real$

function $\mathcal{O}(Request)$
    $m \leftarrow GetData(UID, scope)$
    $k \leftarrow_R$ `Keys`
    $salt \leftarrow \{0,1\}^{saltLen}$
    return $E_k(Request \| m \| salt)$

Algorithm 9 Rand$

function $\mathcal{O}(Request)$
    $m \leftarrow GetData(UID, scope)$
    $k \leftarrow_R$ `Keys`
    $r \leftarrow \{0,1\}^{|Request|+|m|+saltLen}$
    return $E_k(r)$

Распишем преимущество противника следующим образом:

$$\mathbb{P}[Real(\mathcal{A}) \to 1] - \mathbb{P}[Rand(\mathcal{A}) \to 1] = \\ = \Big(\mathbb{P}[Real(\mathcal{A}) \to 1] - \mathbb{P}[Real\$(\mathcal{A}) \to 1]\Big)+ \\ \Big(\mathbb{P}[Real\$(\mathcal{A}) \to 1] - \mathbb{P}[Rand\$(\mathcal{A}) \to 1]\Big)+ \\ +\Big(\mathbb{P}[Rand\$(\mathcal{A}) \to 1] - \mathbb{P}[Rand(\mathcal{A}) \to 1]\Big) \quad (10)$$

1) Первое слагаемое оценивается сверху величиной
$$Adv^{MDDH}(t+q\mu,q)$$
В первом слагаемом мы переходим от ключей, вырабатываемых по протоколу VKO к случайным ключам шифрования. Если есть противник, который хорошо различает два указанных режима, то мы можем использовать его для решения задачи $MDDH$. Если нам дан оракул $\mathcal{O}$ из задачи MDDH, то нам достаточно промоделировать все условия для противника, различающего режимы. Для имитации режима шифрования требуется не более $q\mu$ зашифрований блоков (считаем, что зашифрование одного блока информации является элементарной операцией).
2) Второе слагаемое оценивается сверху величиной:
$$q \cdot Adv_{SE}^{ROR-CPA}(t+q\cdot(2\mu+1),1,\mu),$$
поскольку мы можем вновь рассмотреть гибридный аргумент: переход от $\xi_i = E_k(m)$ к $\eta_i = E_k(r)$ со случайно выбранным ключом k. Все остальные координаты случайного вектора мы можем смоделировать. Для этого необходимо:
   a) Выбрать ключ $k \in$ `Keys`;
   b) Выбрать случайные r соответствующей длины;
   c) Зашифровать m или r по соответствующей схеме шифрования.

   Время на моделирование вектора можно оценить сверху как:
$$\underbrace{q}_{\text{генерация ключей}} + \underbrace{q\mu}_{\text{генерация случайных } r} + \underbrace{q\mu}_{\text{зашифрование}}.$$
3) В третьем слагаемом мы вновь возвращаемся от случайных ключей к ключам, вырабатываемым по протоколу VKO. Это слагаемое оценивается сверху величиной:
$$Adv^{MDDH}(t+2q\mu,q),$$
поскольку если есть противник, хорошо различающий два режима ($Rand\$$ и $Rand$), то мы можем использовать этого противника для решения задачи MDDH. Для моделирования необходимо $q\mu$ операций на генерацию r соответствующей длины и $q\mu$ операций на зашифрование.

Отсюда получаем необходимую оценку. □

Заменяя оценку в задаче **MDDH** на оценку задачи **DDH**, получим итоговую оценку:

Утверждение 3. *Преимущество противника в задаче различения ответов может быть ограничено сверху как:*

$$Adv^{ROR-Reply}(t,q,\mu) \leq \\ \leq 2q \cdot Adv^{DDH}(t + q \cdot (4 + 2\mu)) + \\ + q \cdot Adv_{SE}^{ROR-CPA}(t + q \cdot (2\mu + 1), 1, \mu) \quad (11)$$

d) Что будет, если нарушено одно из свойств: Свойство **ROR-CPA** формулируется в условиях равновероятности ключа. Вообще говоря, ключ K_{UI} может быть неравновероятным при известных открытых ключах пользователя и инспектора. Для того, чтобы заменить ключ K_{UI} на равновероятный, требуется **DDH**-предположение.

Даже при условии, что ключ выбран равновероятно, сам режим шифрования может быть слабым и допускать утечку некоторой информации об открытом тексте. Чтобы исключить это явление, требуется **ROR-CPA** стойкость режима шифрования.

Если выработанный в протоколе Диффи-Хеллмана ключ не является равновероятным, то мы получаем некоторое апостериорное распределение на ключах; данная неравномерность может быть использована противником для сокращения числа перебираемых ключей.

Если ключ является равновероятным, но режим шифрования допускает утечку информации (не выполнено свойство **ROR-CPA**), то противник может (теоретически) извлекать информацию о персональных данных пользователя, изучая его ответы на запросы сервиса.

F. Неподделываемость блоба (blob unforgeability)

a) Требование: противник не должен иметь возможность со значимой вероятностью подделывать блоб пользователя

$$[y, UID, Reply]_{sgn(U)}$$

b) Чем обеспечивается: **EUF-CMA** стойкость схемы подписи (в том числе, устойчивость к коллизиям хэш-функции $H(\cdot)$, используемой в подписи, если она имеется).

c) Формализация требования: Противник имеет право подавать оракулу $\mathcal{O}$ запросы вида:

$$Req_i = (SrcID, UID, scope, ttl)$$

Оракул по $scope$ определяет Инспектора для проверки данных и проводит с ним протокол выработки общего секретного ключа $k = K_{UI}$. Данная процедура происходит при каждом новом запросе.

На запрос противника оракул $\mathcal{O}$ формирует $blob$:

$$blob = [y_i, UID, E_k(Req_i, SecData_i, salt_i)]_{sgn(U)}$$

и возвращает результат противнику.

Изучая ответы оракула, противник должен сформировать валидный блоб от имени пользователя U, то есть предъявить:

$$blob^* = \\ = [y, UID, E_k(Req^*, SecData^*, salt^*)]_{sgn(U)}$$

где $Req^* \neq Req_i, i = 1, \ldots, q$, то есть противник получает блоб на запрос, которого ещё не было.

Формально определим экспериментатора $Blob$, который будет запоминать все полученные от противника запросы.

Оракул $\mathcal{O}$ будет реализовывать формирование блоба. Противник задает некоторое количество запросов к оракулу $\mathcal{O}$, после чего выдает подделанный блоб $blob^*$.

Экспериментатор проверяет, что:

1) Сертификат открытого ключа пользователя верный;
2) Подпись верифицируема;
3) Запрос Req не был запрошен ранее;

Мы несколько упрощаем задачу противнику, поскольку для верификации блоба дополнительно требуется, чтобы UID, указанный в $Request$, совпадал с ID подписи, а также чтобы персональные данные пользователя совпадали с указанными.

Для того, чтобы блоб прошёл проверку, достаточно, чтобы произошло хотя бы одно следующих двух событий:

A_1**:** Противнику удалось подделать сертификат публичного ключа пользователя под своим личным ключом. В таком случае противник может подделывать любые блобы.

A_2**:** Противнику удалось подделать подпись под новым блобом.

И в первом, и во втором случае противник решает задачу **EUF-CMA** для схемы подписи, используемой в протоколе.

Таким образом, верно следующее утверждение:

Утверждение 4. *Вероятность подделки блоба может быть оценена сверху как:*

$$\mathbb{P}[Blob(\mathcal{A}) \to 1] \leq \mathbb{P}[A_1] + \mathbb{P}[A_2] \leq \\ \leq Adv_{Sert}^{\textbf{EUF-CMA}}(q_{usr}, t) + Adv_{User}^{\textbf{EUF-CMA}}(q_{blob}, t),$$

Algorithm 10 Blob

function INIT
  $requests := [\,]$
function $\mathcal{O}(Req)$
  $requests = requests \cup Req$
  $m \leftarrow GetData(UID, scope)$
  $InspID \leftarrow GetID(scope)$
  $k \leftarrow VKO(K_U^{priv}, K_I^{publ})$
  $salt \leftarrow \{0,1\}^{saltLen}$
  $Reply \leftarrow E_k(Req \| m \| salt)$
  $c \leftarrow (K_U^{publ}, UID, Reply)$
  $\sigma \leftarrow Sign_U(c)$
  return (c, σ)
function FIN($blob^* = [y, UID, Reply]_\sigma$)
  if $!(SertCheck(\sigma))$ **then**
    return 0
  $c \leftarrow (y, UID, Reply)$
  if $!(Verify(c, \sigma))$ **then**
    return 0
  $k \leftarrow VKO(K_U^{publ}, K_I^{priv})$
  $(Req \| SecData \| salt) \leftarrow D_k(Reply)$
  if $Req \in requests$ **then**
    return 0
  return 1

где использованы обозначения: q_{usr} *— количество пользователей, заверенных в* ***CA****;* q_{blob} *— количество блобов, подписанных пользователем;* t *— время, затраченное противником на решение соответствующей задачи* ***EUF-CMA****.*

d) Что будет, если нарушено свойство: Если у функции $H(\cdot)$ легко найти коллизию, то вероятно, что задача поиска второго прообраза также не является сложной.

В таком случае противник может перенести подпись с одного блоба пользователя на другой сгенерированный блоб. Пользователь не сможет оспорить факт предоставления фальшивого блоба, поскольку под ним будет стоять его подпись. Сервис не сможет убедиться, что именно пользователь U предоставил персональные данные.

Даже при условии, что функция $H(\cdot)$ устойчива к коллизиям, сама схема подписи может быть слабой, что позволит противнику подделать подпись под совершенно другим блобом.

G. Неподделываемость запроса (request unforgeability)

a) Требование: противник не должен уметь подделывать запрос Сервиса

$$Request = [SrcID, UID, scope, ttl]_{sgn(S)}$$

b) Чем обеспечивается: **EUF-CMA**-стойкость схемы подписи (в том числе, устойчивость к коллизиям хэш-функции $H(\cdot)$, используемой в подписи, если она имеется).

c) Формализация требования: По аналогии с предыдущей задачей, рассмотрим следующую формализацию:

Algorithm 11 Request

function INIT
  $requests := [\,]$
function $\mathcal{O}(UID, scope, ttl)$
  $R \leftarrow (SrcID, UID, scope, ttl)$
  $requests.add(R)$
  $\sigma \leftarrow Sign_{Src}(R)$
  return (R, σ)
function FIN(R, σ)
  if $!(SertCheck(\sigma))$ **then**
    return 0
  return $Verify(c, \sigma)$

Для того, чтобы запрос (Request) прошёл проверку, достаточно, чтобы произошло хотя бы одно следующих двух событий:

A_1**:** Противнику удалось подделать сертификат публичного ключа сервиса под своим ключом. В таком случае противник может подделывать любые запросы.

A_2**:** Противнику удалось подделать подпись под новым запросом.

И в первом, и во втором случае противник решает задачу **EUF-CMA** для схемы подписи, используемой в протоколе.

Таким образом, верно следующее утверждение:

Утверждение 5. *Вероятность подделки запроса оценивается сверху как:*

$$\mathbb{P}[Request(\mathcal{A}) \to 1] \le \mathbb{P}[A_1] + \mathbb{P}[A_2] \le$$
$$\le Adv_{Sert}^{\textbf{EUF-CMA}}(q_{usr}, t) + Adv_{src}^{\textbf{EUF-CMA}}(q_{src}, t),$$

где использованы обозначения: q_{usr} *— количество пользователей, заверенных в* ***CA****;* q_{src} *— количество запросов, подписанных сервисом;* t *— время, затраченное противником на решение соответствующей задачи* ***EUF-CMA****.*

H. Неподделываемость ответа инспектора

a) Требование: противник не должен уметь подделывать ответ инспектора

$$check = [yes/no, ttl, blob]_{sgn(I)}$$

b) Чем обеспечивается: **EUF-CMA** стойкость схемы подписи (в том числе, устойчивость к коллизиям хэш-функции $H(\cdot)$, используемой в подписи, если она имеется).

Аналогично двум предыдущим случаям, можно доказать следующее утверждение:

Утверждение 6. *Вероятность подделки ответа инспектора оценивается сверху как:*

$$\mathbb{P}[Insp(\mathcal{A}) \to 1] \leq$$
$$\leq Adv_{sert}^{\textbf{EUF-CMA}}(q_{usr}, t) + Adv_{insp}^{\textbf{EUF-CMA}}(q_{insp}, t),$$

где использованы обозначения: q_{usr} *— количество пользователей, заверенных в* ***CA****;* q_{insp} *— количество запросов, подписанных инспектором;* t *— время, затраченное противником на решение соответствующей задачи* ***EUF-CMA****.*

c) Что будет, если нарушено одно из свойств: в этом случае у противника (как уже описано выше) есть возможность подделывать ответ Инспектора при проверке персональных данных. Тогда он, в частности, может принимать некорректные персональные данные (в сговоре с Пользователем), продлевать по своему усмотрению срок действия персональных данных (ttl), предоставлять некорректные персональные данные.

I. Выводы

Собирая вместе результаты утверждений 3, 5, 4, 6, получим, что вероятность любого из нежелательных событий (нарушения работы протокола) не превосходит максимума из следующих величин:

$$Adv^{ROR-Reply}(\mathcal{A}), \mathbb{P}[Blob(\mathcal{A}) \to 1],$$
$$\mathbb{P}[Request(\mathcal{A}) \to 1], \mathbb{P}[Insp(\mathcal{A}) \to 1],$$

который, в свою очередь, может быть оценен сверху как сумма данных величин (в силу их неотрицательности):

$$Adv^{ROR-Reply}(\mathcal{A}) + \mathbb{P}[Blob(\mathcal{A}) \to 1] +$$
$$+\mathbb{P}[Request(\mathcal{A}) \to 1] + \mathbb{P}[Insp(\mathcal{A}) \to 1],$$

откуда следует исходная теорема 1

Теорема 1.

Вероятность взлома протокола может быть оценена сверху как:

$$2q \cdot Adv^{DDH}(t + q \cdot (4 + 2\mu)) +$$
$$+q \cdot Adv_{SE}^{ROR-CPA}(t + q \cdot (2\mu + 1), 1, \mu) +$$
$$+3 \cdot Adv_{Sert}^{\textbf{EUF-CMA}}(q_{usr}, t) +$$
$$+Adv_{insp}^{\textbf{EUF-CMA}}(q_{insp}, t) + Adv_{User}^{\textbf{EUF-CMA}}(q_{blob}, t) +$$
$$+Adv_{src}^{\textbf{EUF-CMA}}(q_{src}, t)$$

где использованы обозначения:

q_{usr} *— количество пользователей, заверенных в* ***CA****;*

q_{blob} *— количество блобов, подписанных пользователем;*

q_{src} *— количество запросов, подписанных сервисом;*

q_{insp} *— количество запросов, подписанных инспектором;*

q *— количество запросов* Req *противника на зашифрование;*

μ *— длина наибольшего запрошенного* $Request \| SecData \| salt$;

t *— время, затраченное противником;*

Библиография

[1] The Kerberos Network Authentication Service (V5) : RFC : 4120 / RFC Editor ; Executor: C. Neuman, T. Yu, S. Hartman, K. Raeburn : 2005. — July. — URL: http://www.rfc-editor.org/rfc/rfc4120.txt.

[2] The OAuth 2.0 Authorization Framework : RFC : 6749 / RFC Editor ; Executor: D. Hardt : 2012. — October. — URL: http://www.rfc-editor.org/rfc/rfc6749.txt.

[3] Оганесян А. Самые значительные утечки данных в 2018 году (часть первая). — URL: https://habr.com/ru/company/devicelockdlp/blog/432354/. дата обращения: 07.11.2019.

[4] А. Оганесян. Самые значительные утечки данных в 2018 году (часть вторая). — URL: https://habr.com/ru/company/devicelockdlp/blog/434000/. дата обращения: 07.11.2019.

[5] InfoWatch. За 12 лет утекло более 30 млрд записей персональных данных. — URL: https://www.infowatch.ru/resources/analytics/digest/15281. дата обращения: 29.11.2019.

[6] Н. Ильина, А. Урманцева. Из базы вон: данные о клиентах банков из топ-20 продают в Telegram. — URL: https://iz.ru/906688/natalia-ilina-anna-urmantceva/iz-bazy-von-dannye-o-klientakh-bankov-iz-top-20-prodaiut-v-telegram. дата обращения: 29.11.2019.

[7] Федеральный закон «О персональных данных» от 27.07.2006 n 152-ФЗ. — URL: http://www.consultant.ru/document/cons_doc_LAW_61801/.

[8] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance). — дата обращения: 07.11.2019. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679.

[9] Samonte M. Google v CNIL Case C-507/17: The Territorial Scope of the Right to be Forgotten Under EU Law. — 2019. — URL: https://europeanlawblog.eu/2019/10/29/google-v-cnil-case-c-507-17-the-territorial-scope-of-the-right-to-be-forgotten-under-eu-law/. дата обращения: 07.11.2019.
[10] GDPR Fines and Penalties. — URL: https://www.nathantrust.com/gdpr-fines-penalties. дата обращения: 07.11.2019.
[11] ФСБ предупредила о риске утечек из создающейся единой базы персональных данных. — URL: https://www.kommersant.ru/doc/4156290. дата обращения: 17.11.2019.
[12] The Transport Layer Security (TLS) Protocol Version 1.3 : RFC : 8446 / RFC Editor ; Executor: E. Rescorla : 2018. — August.
[13] Информационная технология. Криптографическая защита информации. Криптографические алгоритмы, сопутствующие применению алгоритмов электронной цифровой подписи и функции хеширования. ГОСТ Р 50.1.113-2016. — дата обращения: 07.11.2019. URL: https://tc26.ru/standard/rs/P%2050.1.113-2016.pdf.
[14] Межгосударственный стандарт ГОСТ 34.10.2018 Информационная технология (ИТ). Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи.
[15] Межгосударственный стандарт ГОСТ 34.12-2018 Информационная технология (ИТ). Криптографическая защита информации. Блочные шифры.
[16] Katz Jonathan, Lindell Yehuda. Introduction to modern cryptography. — Chapman and Hall/CRC, 2014.
[17] Bellare Mihir, Rogaway Phillip. Introduction to modern cryptography // Ucsd Cse. — 2005. — Vol. 207. — P. 207.

Description of personal data exchange protocol: X

V. Belsky, I. Gerasimov, K. Tsaregorodtsev, I. Chizhov

***Abstract*—Personal data exchange and disclosure prevention are widespread problems in our digital world. There are a couple of information technologies embedded in the commercial and government processes. People need to exchange their personal information while using these technologies while services need to process it. And therefore, It is essential to make this exchange is secure. Despite many legal regulations, there are many cases of personal data breaches that lead to undesirable consequences. Reasons for personal data leakage may be adversary attack, data administration error or service corruption. At the same time, creating complex service interaction and multilayer information security may lead to many inconveniences for the user. Personal data exchange protocol has the following tasks: participant's data transfer, ensuring information security, providing participants with trust in each other and ensuring service availability. In this paper, we represent a personal data exchange protocol called X. The main idea is to provide personal data encryption on the user side and thus to prevent personal data disclosure and publication. This approach allows us to transfer personal data from user to service only in the form of an encrypted data packet — blob. Each blob can be validated and certified by a personal data inspector who had approved user's information. It can be any government department or a commercial organization, for example, passport issuing authority, banks, etc. It implies that we can provide several key features for personal data exchange. A requesting service cannot publish the user personal data. It still can perform a validation protocol with an inspector to validate user data. We do not depend on service data administration infrastructure and do not complicate the inspector's processes by adding additional information about the personal data request. The personal data package has a link between the personal data owner and a service request. Each blob is generated for a single request and has a time limit for a provided encrypted personal data. After this limit, the service can not use a received package. The user cannot provide invalid personal data or use the personal data of another person. We don't restrict specified cryptographic algorithms usage The X protocol can be implemented with any encryption, digital signature, key generation algorithms which are secure in our adversary model. For protocol description, Russian standardized cryptographic protocols are used. The paper also contains several useful examples of how the X protocol can be implemented in real information systems.**

***Keywords*—X, personal data, VKO GOST, symmetric cryptography**

References

[1] The Kerberos Network Authentication Service (V5) : RFC : 4120 / RFC Editor ; Executor: C. Neuman, T. Yu, S. Hartman, K. Raeburn : 2005. — July. — URL: http://www.rfc-editor.org/rfc/rfc4120.txt.

[2] The OAuth 2.0 Authorization Framework : RFC : 6749 / RFC Editor ; Executor: D. Hardt : 2012. — October. — URL: http://www.rfc-editor.org/rfc/rfc6749.txt.

[3] A. Oganesyan. Samye znachitel'nye utechki dannykh v 2018 godu (chast' pervaia). — URL: https://habr.com/ru/company/devicelockdlp/blog/432354/. (accessed: 07.11.2019), (in Russian).

[4] A. Oganesyan. Samye znachitel'nye utechki dannykh v 2018 godu (chast' vtoraia). — URL: https://habr.com/ru/company/devicelockdlp/blog/434000/. (accessed: 07.11.2019), (in Russian).

[5] InfoWatch. Za 12 let uteklo bolee 30 mlrd zapisei personal'nykh dannykh. — URL: https://www.infowatch.ru/resources/analytics/digest/15281. (accessed: 07.11.2019), (in Russian).

[6] N. Ilina, A. Urmantseva. Iz bazy von: dannye o klientakh bankov iz top-20 prodaiut v Telegram. — URL: https://iz.ru/906688/natalia-ilina-anna-urmantceva/iz-bazy-von-dannye-o-klientakh-bankov-iz-top-20-prodaiut-v-telegram. (accessed: 29.11.2019), (in Russian).

[7] Federal'nyi zakon «o personal'nykh dannykh» ot 27.07.2006 n 152-fz. — URL: http://www.consultant.ru/document/cons_doc_LAW_61801/.

[8] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance). — (accessed: 07.11.2019). URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679.

[9] Samonte M. Google v CNIL Case C-507/17: The Territorial Scope of the Right to be Forgotten Under EU Law. — 2019. — URL: https://europeanlawblog.eu/2019/10/29/google-v-cnil-case-c-507-17-the-territorial-scope-of-the-right-to-be-forgotten-under-eu-law/. (accessed: 07.11.2019).

[10] GDPR Fines and Penalties. — URL: https://www.nathantrust.com/gdpr-fines-penalties. (accessed: 07.11.2019).

[11] FSB predupredila o riske utechek iz sozdaiushcheisia edinoi bazy personal'nykh dannykh. — URL: https://www.kommersant.ru/doc/4156290. (accessed: 17.11.2019), (in Russian).

[12] The Transport Layer Security (TLS) Protocol Version 1.3 : RFC : 8446 / RFC Editor ; Executor: E. Rescorla : 2018. — August.
[13] Informacionnaja tehnologija. kriptograficheskaja zashhita informacii. kriptograficheskie algoritmy, soputstvujushhie primeneniju algoritmov jelektronnoj cifrovoj podpisi i funkcii heshirovanija. gost r 50.1.113-2016. — (accessed: 07.11.2019), (in Russian). URL: https://tc26.ru/standard/rs/P%2050.1.113-2016.pdf.
[14] Mezhgosudarstvennyi standart gost 34.10.2018 informatsionnaia tekhnologiia (it). kriptograficheskaia zashchita informatsii. protsessy formirovaniia i proverki elektronnoi tsifrovoi podpisi. — (in Russian).
[15] Mezhgosudarstvennyi standart gost 34.12-2018 informatsionnaia tekhnologiia (it). kriptograficheskaia zashchita informatsii. blochnye shifry. — (in Russian).
[16] Katz Jonathan, Lindell Yehuda. Introduction to modern cryptography. — Chapman and Hall/CRC, 2014.
[17] Bellare Mihir, Rogaway Phillip. Introduction to modern cryptography // Ucsd Cse. — 2005. — Vol. 207. — P. 207.