МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

УДК 004.021

СХЕМА ГРУППОВОЙ АУТЕНТИФИКАЦИИ НА ОСНОВЕ ДОКАЗАТЕЛЬСТВА С НУЛЕВЫМ РАЗГЛАШЕНИЕМ

Е. А. Шляхтина, Д. Ю. Гамаюнов

Московский государственный университет имени М. В. Ломоносова, г. Москва, Россия

Рассматривается проблема взаимной аутентификации пользователей в децентрализованных системах обмена сообщениями в отсутствие доверенной третьей стороны. Предложен алгоритм взаимной аутентификации пользователей групп на основе доказательства с нулевым разглашением. Алгоритм позволяет аутентифицировать пользователей децентрализованной сети без установки общего секрета по стороннему каналу, опираясь на существующие в сети цепочки доверия. В основе метода лежит протокол демократичной групповой подписи DGS и алгоритм выработки общего ключа для больших и динамических групп CCEGK. Проведены анализ безопасности, устойчивости схемы аутентификации, в том числе к атаке Сивиллы, и оценка сложности предлагаемого алгоритма. Алгоритм реализован на модельном децентрализованном приложении на основе P2P топологии Chord, с помощью модельной реализации сделаны оценки накладных расходов схемы аутентификации и времени сходимости для некоторых частных случаев конфигураций групп пользователей и начальных цепочек доверия.

Ключевые слова: *аутентификация, доказательство с нулевым разглашением, децентрализованные коммуникации.*

DOI 10.17223/20710410/51/3

GROUP AUTHENTICATION SCHEME BASED ON ZERO-KNOWLEDGE PROOF

E. A. Shliakhtina, D. Y. Gamayunov

Lomonosov Moscow State University, Moscow, Russia

E-mail: ea.shlyakhtina@gmail.com, gamajun@seclab.cs.msu.su

In this paper, we address the problem of mutual authentication in user groups in decentralized messaging systems without trusted third party. We propose a mutual authentication algorithm for groups using zero-knowledge proof. Using the algorithm, which is based on trust chains existing in decentralized network, users are able to authenticate each other without establishing a shared secret over side channel. The proposed algorithm is based on Democratic Group Signature protocol (DGS) and Communication-Computation Efficient Group Key algorithm for large and dynamic groups (CCEGK). We have performed security analysis of the proposed mutual authentication scheme against several attacks including Sybil attack and have made

complexity estimation for the algorithm. The algorithm is implemented in an experimental P2P group messaging application, and using this implementation we estimate overhead of the authentication scheme and convergence time for several initial configurations of user groups and trust chains.

Keywords: *authentication, zero-knowledge proof, decentralized communications.*

Введение

В настоящее время значительно выросла роль систем мгновенного обмена сообщениями и требования к обеспечению конфиденциальности таких систем коммуникаций. При этом все коммерческие системы, доступные в Интернете, представляют собой централизованнные системы с доверенной третьей стороной, в роли которой выступает поставщик сервиса. В то же время существует значительный исследовательский интерес к одноранговым сетям для реализации систем обмена сообщениями, в которых обеспечивается равноправие всех участников сети и не требуется доверенная третья сторона. В таких системах участник сети одновременно является клиентом и выполняет функции сервера для формирования распределённой децентрализованной топологии и обеспечения возможности пересылки сообщений других участников сети. Среди достоинств таких сетей — гибкость при изменении структуры сети, надёжность и отказоустойчивость. Принципы децентрализации хорошо подходят для создания мессенджеров — приложений для обмена мгновенными сообщениями.

В работе [1] предложена схема многопользовательских защищённых коммуникаций на основе протокола multi-party Off-The-Record (mpOTR) [2] со свойствами секретности будущих сообщений, отказуемости и согласованности текста переписки. Протокол основан на криптосистеме с открытым ключом. Данная схема была использована при разработке криптографического группового чата [3, 4]. При реализации чата был также использован алгоритм аутентификации долговременных открытых ключей пользователей [5] на основе протокола Социалистов-Миллионеров [6], являющегося протоколом с нулевым разглашением. Аутентификация открытых ключей служит защитой от атаки «человек в середине». Протокол Социалистов-Миллионеров позволяет сравнивать общий секрет двух пользователей, содержащий отпечатки[1] их открытых ключей и секретную фразу, о которой участники договариваются по стороннему каналу связи. Однако не всегда есть возможность безопасной установки общего секрета или задания вопроса, на который собеседник, и только он, точно знает ответ.

В данной работе предлагается метод аутентификации открытых ключей пользователей без использования общего секрета, установленного по стороннему каналу связи, обрабатывающий данные об уже существующих цепочках доверия в сети. Метод применим в системе со свойствами отказуемости. В его основе лежит механизм групповой подписи с нулевым разглашением [7], позволяющий членам аутентифицированных групп подписывать сообщения от их имени и обеспечивающий невозможность подделки групповой подписи участниками, не являющимися членами группы. Если пользователь получит информацию об открытом ключе другого пользователя, подтверждённую цепочкой доверенных участников, связывающих их, то он может доверять этой информации.

[1]Применяя криптографическую хеш-функцию к открытому ключу, можно получить отпечаток ключа, идентифицирующий его.

1. Модель

Сетевая модель: децентрализованная топология сети, каждый участник сети обладает уникальным идентификатором id, парой открытый/секретный ключ (pk, sk).

Модель нарушителя: участник, имеющий возможность перехватывать трафик и контролировать узлы сети. Цель нарушителя — выдать себя за другого человека, подменив соответствующие открытые ключи.

Модельное приложение: децентрализованное приложение для защищённого обмена сообщениями в групповых конференциях, реализованное с помощью браузерной технологии WebRTC [8] (Web Real Time Communication), позволяющей пользователям установить P2P (Peer-to-Peer)-соединение. Пиринговая сеть основана на распределённой хэш-таблице Chord [9], размещённой в узлах сети. На основе этого протокола строится одноранговая сеть с топологией «кольцо», которая используется в качестве P2P-транспорта для чата. Chord служит для организации эффективного доступа к узлам распределённой системы и хранения информации. Безопасность групповых коммуникаций обеспечивается протоколом mpOTR [2], в основе которого лежит криптосистема с открытым ключом. В реализации используется система защищённых групповых коммуникаций, предложенная в [1] и обладающая свойствами конфиденциальности, секретности будущих сообщений, аутентичности, отказуемости, целостности передачи данных и согласованности текста переписки.

2. Краткое описание используемых протоколов

Механизм групповой подписи позволяет членам группы анонимно подписываться от имени группы так, что любой проверяющий может убедиться в том, что документ был подписан участником группы. Особенностью групповой подписи является наличие менеджера группы, который может раскрывать личность выпустившего ту или иную подпись. В обязанности менеджера входит контроль всех динамических изменений в группе (например, добавление или исключение участников), а потому он должен поддерживать группу на протяжении всего её существования. Кроме того, остальные участники должны ему доверять. Очевидно, наличие такого доверенного участника является узким местом в схемах групповых подписей. Для повышения надёжности в некоторых из них существуют модификации, позволяющие разделить обязанности по добавлению новых участников и раскрытию личности подписавшегося между двумя менеджерами, однако уровень доверия к ним остаётся по-прежнему высоким.

В 2006 г. опубликована работа [7] Марка Манулиса, в которой представлена схема демократичной групповой подписи DGS, где роль доверенного менеджера группы отсутствует. DGS позволяет инициализировать группу, принимать решение о принятии нового члена в группу и об исключении старых коллективно всеми участниками, а возможность выявить личность подписавшего появляется у всех членов группы. Вариант групповой подписи без такого элемента централизации, как доверенный менеджер, является наиболее естественным для реализации в условиях одноранговой сети.

Предполагается выполнение следующих условий:

1) члены группы не разглашают данные, которые могут быть использованы посторонними для раскрытия издателя подписи;
2) в группе есть хотя бы один честный пользователь;
3) пользователи аутентифицируют свои сообщения в процессе коммуникации.

Рассматривается пассивный противник, не являющийся членом группы, так как в результате работы протокола все участники будут обладать групповым секретом.

В основе протокола лежат следующие блоки:

— протокол выработки общего секрета группы CGKA (Contributory Group Key Agreement protocol);
— подпись знания SK (Signature of Knowledge).

Прежде чем перейти к схеме групповой подписи, приведём описание этих блоков.

2.1. Протокол выработки общего секрета

$$\mathbf{CGKA} = \{\mathbf{Setup}, (\mathbf{Join}_i, \mathbf{Join}_u), \mathbf{Leave}\}$$

Setup — интерактивный алгоритм, исполняющийся каждым из n пользователей, желающих выработать общий секрет; инициализирует группу.

Вход: индивидуальный секрет[2] i-го пользователя k_i, соответствующий вклад z_i.

Выход: групповой секрет k_G.

В результате взаимодействия участников у каждого формируется множество вкладов остальных пользователей $Z = \{z_j : j \in \{1, \ldots, n\}, j \neq i\}$.

$(\mathbf{Join}_i, \mathbf{Join}_u)$ — пара интерактивных алгоритмов, выполняющихся группой и новым членом группы соответственно для добавления нового участника в группу.

Вход $\mathbf{Join}_i$: индивидуальный секрет i-го пользователя k_i, вклад нового пользователя z_u, структура группы.

Вход $\mathbf{Join}_u$: индивидуальный секрет нового пользователя k_u, множество вкладов участников Z, структура группы.

Выход $(\mathbf{Join}_i, \mathbf{Join}_u)$: обновлённое множество вкладов пользователей Z, новый групповой секрет k_G.

Leave — интерактивный алгоритм, выполняющийся между $(n-1)$ остающимися участниками для исключения j-го участника из группы. В результате изменяются индивидуальные секреты некоторых пользователей, обновляется список вкладов пользователей, структура группы и групповой секрет k_G.

Вход: индивидуальный секрет участника k_i, вклад покидающего группу z_j, структура группы.

Выход: обновлённые список вкладов пользователей Z, структура группы и групповой секрет k_G.

В работе [7] не фиксируется определённый протокол выработки общего секрета, но требуется наличие следующих ключевых свойств у протокола:

— для пассивного противника вычислительно сложно найти групповой секрет;
— для пассивного противника вычислительно сложно отличить случайные биты от битов группового секрета;
— секретность будущих сообщений (forward secrecy) — обладая некоторым множеством старых групповых секретов, пассивный противник не сможет вычислить новые;
— секретность прошлых сообщений (backward secrecy) — обладая некоторым множеством текущих групповых секретов, пассивный противник не сможет вычислить старые;
— независимость групповых секретов.

[2]Каждый пользователь в группе имеет индивидуальный секрет, на основе совокупности индивидуальных секретов участников группы вырабатывается общий секрет.

Кроме того, определён вид вкладов участников: $z_i = g^{k_i}$. Здесь g — образующий элемент мультипликативной группы $\mathbb{Z}_p^*$, где p — большое простое число.

В качестве протокола выработки общего секрета в [7] рекомендуется протокол TGDH (Tree-based Group Diffe — Hellman), представляющий собой процедуру генерации общего секрета на основе протокола Диффи — Хеллмана и древовидной структуры группы. Древовидная структура позволяет эффективно пересчитывать общий секрет при динамических событиях, что важно в случае больших групп. Протоколами, основанными на этом принципе и удовлетворяющими требованиям, являются EGK [10], STR [11, 12], CCEGK [13]. Последний является улучшенной версией TGDH с точки зрения вычислительной и коммуникационной сложности, что показано в работе [13]. Поэтому именно этот протокол выбран для построения схемы аутентификации.

2.2. Подпись знания

Подпись знания [14] представляет собой неинтерактивное доказательство знания некоторого секрета с нулевым разглашением, зависящее от сообщения. Неинтерактивным оно становится благодаря эвристическому методу Фиата — Шамира [15].

$$\mathbf{SK} = \{\mathbf{SKSig}, \mathbf{SKVer}\}$$

SKSig — вероятностный алгоритм генерации подписи; **SKVer** — детерминированный алгоритм ее проверки.

Определение 1. Предположение DL (Discrete Logarithm assumption) о сложности дискретного логарифмирования. Пусть G — циклическая группа с образующим элементом g. Не существует вероятностного полиномиального алгоритма A, который с непренебрежимо малой вероятностью вычисляет $\log_g(g^a)$, получая на вход (G, g, g^a), где $a \in \mathbb{Z}_{\mathrm{ord}(G)}$.

Предположение DL выполняется в G, если $Pr[A(G, g, g^a) = a] \leqslant negl(n)$, где $n = \|\mathrm{ord}(G)\|$. Здесь $\|x\|$ — длина записи двоичного представления числа x; $negl$ — пренебрежимо малая функция.

Определение 2. Пусть G — циклическая группа и выполняется предположение DL. Тогда

$$(c, s) \in \{0,1\}^k \times \mathbb{Z}_{\mathrm{ord}(G)}, \quad (c, s) = SK[(\alpha) : y = g^\alpha](m)$$

есть подпись знания дискретного логарифма элемента $y \in G$ по основанию g для сообщения m, если $c = H(m||y||g||g^s y^c)$, где $H : \{0,1\}^* \to \{0,1\}^k$ — криптографическая хэш-функция.

Алгоритм её вычисления при знании $x = \log_g(y)$:

1) $r \in_R \mathbb{Z}^*_{\mathrm{ord}(G)}$;
2) $c = H(m||y||g||g^r)$;
3) $s = r - cx$.

Этот подход позволяет конструировать подписи знания более сложных отношений. В [7] для генерации групповой подписи используются такие подписи знания:

1) $SK[(\alpha_i, \beta) : y = g_1^\beta g_2^{\alpha_i} \wedge (z_1 = g_2^{\alpha_1} \vee \ldots \vee z_n = g_2^{\alpha_n})](m)$ — подпись знания дискретного логарифма по основанию g_2 одного значения множества $\{z_1, \ldots, z_n\}$ и равенства этой величины экспоненте g_2-составляющей y;
2) $SK[(\alpha, \beta) : y_1 = g_2^\beta \wedge y_2 = g_1^\beta g_2^\alpha](m)$ — подпись знания представления y_2 по основаниям g_1 и g_2 и равенства дискретного логарифма y_1 по основанию g_2 и экспоненты g_1-составляющей величины y_2.

2.3. Схема групповой подписи

$$\mathbf{DGS} = \{\mathbf{Setup}, \mathbf{Join}, \mathbf{Leave}, \mathbf{Sign}, \mathbf{Verify}, \mathbf{Trace}, \mathbf{TraceVerify}\}$$

Введём следующие обозначения:

- t — счётчик, инициализированный нулём; нужен для отслеживания конфигурации группы и увеличивается при любом динамическом изменении. Все параметры групповой подписи связаны со счётчиком;
- $Y_{[t]}$ — открытый ключ группы, $Y_{[t]} = (g^{\hat{x}_{[t]}}, Z_{[t]}) = (\hat{y}_{[t]}, Z_{[t]})$, где $Z_{[t]} = \{z_{1[t]}, \ldots, z_{n[t]}\}$ — множество вкладов всех участников;
- $x_{i[t]}$ — индивидуальный секрет i-го члена группы;
- $\hat{x}_{[t]}$ — общий секрет группы.

Алгоритм **Setup** выполняется всеми участниками для формирования группы. Счётчик t инициализируется нулём. Пользователи устанавливают индивидуальные секреты $x_{i[0]} \in_R \mathbb{Z}_{p-1}$ и вычисляют свои вклады $z_{i[0]}$. Далее выполняется алгоритм **CGKA**.**Setup**$(x_{i[0]}, z_{i[0]})$ для формирования структуры группы и выработки группового секрета $\hat{x}_{[0]} \in \mathbb{Z}_{p-1}$. Из полученной информации создаётся открытый ключ группы $Y_{[t]}$.

Алгоритм **Join** позволяет новому участнику U присоединиться к группе. U устанавливает свой секрет $x_{u[t+1]} \in_R \mathbb{Z}_{p-1}$ и вычисляет соответствующий вклад $z_{u[t+1]}$, где t — текущий счётчик конфигурации группы. Затем новый участник и группа выполняют алгоритмы протокола выработки группового секрета **CGKA**.**Join**$_u(x_{u[t+1]})$, **CGKA**.**Join**$_i(z_{u[t+1]})$ соответственно. В результате обновляется структура группы и формируется новый групповой секрет.

Ситуация выхода участника из группы обрабатывается алгоритмом **Leave** с помощью алгоритма **CGKA**.**Leave**. Все оставшиеся участники обновляют структуру группы. Один из участников группы должен изменить свой секрет, пересчитать групповой секрет и предоставить данные остальным членам группы для вычисления группового секрета и обновления открытого ключа группы.

Алгоритм **Sign** генерирует групповую подпись $\sigma = (\widetilde{g}, \widetilde{y}, S)$ сообщения $m \in \{0,1\}^*$, где

$$S = SK[(\alpha_i, \beta) : \widetilde{g} = g^{\beta} \wedge \widetilde{y} = \hat{y}_{[t]}^{\beta} g^{\alpha_i} \wedge (z_{1[t]} = g^{\alpha_1} \vee \ldots \vee z_{n[t]} = g^{\alpha_n})](m).$$

Данная подпись является комбинацией подписей знания, описанных в п. 2.2. Тем самым предъявляющий подпись доказывает, что пара $(\widetilde{g}, \widetilde{y})$ шифрует вклад $z_{i[t]}$ в множестве вкладов $Z_{[t]}$, не раскрывая его, а знание дискретного логарифма, используемого для вычисления $z_{i[t]}$, показывает, что подписавшийся действительно владелец вклада $z_{i[t]}$.

Алгоритм **Verify** реализует проверку подписи σ. Для его работы необходим открытый ключ группы $Y_{[t]}$.

Для раскрытия личности подписавшего используется алгоритм **Trace**. На первом этапе осуществляется проверка подписи сообщения с помощью алгоритма **Verify**. Далее с помощью группового секрета $\hat{x}_{[t]}$ выделяется вклад подписавшего сообщение пользователя $z_{i[t]}$, который идентифицирует внутри группы личность подписавшегося. Также на этом шаге формируется доказательство того, что идентификация личности была проведена корректно, т. е. с использованием группового секрета.

Алгоритм **VerifyTrace** реализует проверку корректности результатов работы функции **Trace**; эта процедура может быть проведена не членом группы.

В схеме DGS размер групповой подписи и открытого ключа группы линейно зависят от количества членов группы. В настоящее время существуют групповые подписи (например, [14, 16]), где оба эти параметра являются константами, однако это достигается при помощи доверенного менеджера, который обрабатывает все изменения в составе группы и обновляет базу данных, содержащую информацию об участниках, размер которой также линейно зависит от числа участников.

Безопасность DGS основана на предположениях о дискретном логарифмировании DL и о сложности задачи отличия DDH (Decisional Diffie — Hellman assumption). Основными свойствами протокола DGS являются корректность, отслеживаемость, анонимность, несвязываемость, невозможность фальсификации.

3. Описание схемы аутентификации

Приведём описание схемы взаимной аутентификации пользователей. Предполагается существование в сети групп аутентифицированных между собой пользователей, т. е. каждый член группы имеет достоверную информацию об открытых ключах пользователей в этой группе. Протокол, с помощью которого происходила эта аутентификация, не фиксируется. Это может быть метод [5], именно такой вариант использован для реализации в данной работе. Аутентифицированная группа использует протокол групповой подписи DGS для выработки открытого ключа группы. Таким образом, сеть разбивается на множество групп, члены которых могут доказать свою принадлежность к группе третьим лицам, подписывая сообщения от имени группы.

После формирования группы аутентифицированных пользователей один из участников публикует информацию о группе:

$$groupID : (Y, idList, Sig_G(groupID, Y, idList)),$$

где $groupID$ — идентификатор группы; Y — открытый ключ, $idList = \{id_1, \ldots, id_n\}$ — список идентификаторов членов группы; Sig_G — групповая подпись.

В данной реализации этот участник определяется своим положением в текущей структуре группы. В протоколе CCEGK им является участник, соответствующий самому правому листу в дереве группы.

Наличие цепочки из пересекающихся между собой групп, объединяющей пользователей $U_1, \ldots, U_n$, позволяет им использовать предлагаемый метод аутентификации.

3.1. Поиск цепочек групп

Сначала каждый из участников получает из сети данные о существующих аутентифицированных группах. Затем, используя эту информацию, участники ищут пути друг до друга, которые в дальнейшем используются для аутентификации.

Начнём с описания поиска цепочек групп, связывающих двух пользователей. Цепочки либо состоят из одного звена, либо их длина не больше некоторого заранее установленного числа. Поиск цепочек выполняется алгоритмами 1 и 2.

Получая на вход список групп в сети $groups$ и идентификаторы пользователей id_A, id_B, алгоритм 1 ($findChains$) находит цепочки групп, ведущие от A к B: $groupsA$ — список групп, в которых состоит A; $groupsB$ — список групп, в которых состоит B. Если данные списки не пусты, находится список $sharedGroups$, являющийся пересечением списков $groupsA$ и $groupsB$. Если он не пуст, то эти группы являются маршрутом от A к B и вносятся в список $output$ найденных цепочек. Если список $sharedGroups$ пуст, то поочередно перебираются все группы $group$ из списка $groupsA$ и для каждой из них выполняется процедура $findPath$ поиска маршрута от группы $group$ к списку групп $groupsB$.

Алгоритм 1. $findChains$: поиск цепочек групп от A к B

Вход: $groups$ — список групп; id_A; id_B.
Выход: $output$ — список, содержащий все найденные цепочки.

1: $groupsA := \{group \in groups : id_A \in group\}$;
2: $groupsB := \{group \in groups : id_B \in group\}$;
3: $viewedGroups := [\,]$;
4: $output := [\,]$;
5: **Если** $groupsA \neq \varnothing \ \wedge \ groupsB \neq \varnothing$, **то**
6: $sharedGroups := groupsA \cap groupsB$.
7: **Если** $sharedGroups \neq \varnothing$, **то**
8: $output := output \cup sharedGroups$,
9: **иначе**
10: **Для всех** $group \in groupsA$:
11: $currentPath := []$;
12: $findPath(viewedGroups, group, groupsB, currentPath, groups, output)$.
13: **Вернуть** $output$.

Алгоритм 2 ($findPath$) — рекурсивная процедура поиска маршрута от заданной группы $group$ к списку групп $groupsB$. Входные параметры:

— $viewedGroups$ — список уже рассмотренных групп (не рассматриваются в последующих шагах);
— $group$ — исходная группа маршрута;
— $groupsB$ — конечный список групп маршрута;
— $currentPath$ — список групп, а именно маршрут, пройденный от пользователя A на текущий момент времени;
— $groups$ — опубликованные данные обо всех существующих в сети группах;
— $output$ — список найденных маршрутов.

Вычисляем список $neighbors$ — это окружение группы $group$, т. е. те группы, с которыми она пересекается, за исключением уже рассмотренных $viewedGroups$. Обновляем список рассмотренных групп $newViewedGroups$ и пройденный маршрут $path$ от A на текущий момент времени. Перебираем все группы из списка $neighbors$: если группа содержит пользователя B, то она является завершением маршрута от A к B и данный маршрут вносится в список $output$; в противном случае если число звеньев в текущей цепочке не превосходит $maxNum$, то выполняем процедуру $findPath$ поиска маршрута к $groupsB$.

Расширим алгоритм поиска путей между двумя участниками на группу из n участников. Каждый пользователь сортирует список из идентификаторов членов группы $idList$ и циклически сдвигает его так, что его собственный идентификатор оказывается последним в списке.

Определим $subgroups$ как список множеств пользователей, где в одно множество входят пользователи, если алгоритмом 1 найдены цепочки, объединяющие их. Сначала в соответствии с расположением пользователей в списке для U_i $subgroups$ имеет вид

$$subgroups = (\{U_{i+1}\}, \ldots, \{U_n\}, \{U_1\}, \ldots, \{U_{i-1}\}, \{U_i\}).$$

Затем с помощью алгоритма 3 ($search$) каждый участник находит первого пользователя в списке, до которого есть путь из пересекающихся групп, либо определяет

Алгоритм 2. *findPath*: поиск маршрута от группы *group* к списку групп *groupsB*

Вход: *viewedGroups*, *group*, *groupsB*, *currentPath*, *groups*, *output*.

1: $path := currentPath \cup \{group\}$;
2: $neighbors := \{g \in groups \mid g \cap group \neq \varnothing \wedge g \neq group \wedge g \notin viewedGroups\}$;
3: $newViewedGroups := viewedGroups \cup neighbors$.
4: **Для всех** $g \in neighbors$:
5: **Если** $g \in groupsB$, **то**
6: $output := output \cup \{path \cup \{g\}\}$,
7: **иначе**
8: **Если** $|path| < maxNum$, **то**
9: $findPath(newViewedGroups, g, groupsB, path, groups, output)$.

отсутствие связи с остальными участниками. Для хранения идентификаторов пользователей, до которых участнику не удалось найти путь, существует список недостижимых пользователей *unreachableList*. Это позволяет избегать в будущем проверки существования связи с этими пользователями.

Алгоритм 3. *search*: поиск доступного пользователя для U_i

1: **Для всех** $subgroup \in subgroups$:
2: **Если** $U_i \notin subgroup$, **то**
3: **Для всех** $user \in subgroup$:
4: **Если** $user \notin unreachableList$, **то**
5: $chains := findChains(U_i, user, groups)$.
6: **Если** $chains = \varnothing$, **то**
7: $unreachableList := unreachableList \cup \{user\}$,
8: **иначе**
9: **Вернуть** $(user, chains)$.

Далее $U_1, \ldots, U_n$ обмениваются сообщениями с результатами. Получив сообщение, пользователь U_i осуществляет проверку корректности полученной цепочки:

1) все группы из цепочки должны существовать в списке *groups*;
2) если U_i состоит в данной группе, то происходит проверка данных группы, указанных в *groups*;
3) если U_i не состоит в данной группе, то его индентификатор не должен находиться в списке *idList* группы;
4) полученный список должен быть цепочкой пересекающихся групп.

Если проверка пройдена успешно, то подгруппы в *subgroups*, содержащие пользователей, связанных цепочками, объединяются в одну. Так, если между всеми пользователями $U_1, \ldots, U_n$ существует связь через группы, найденная за установленное количество шагов, то понадобится один запуск описанной процедуры и *subgroups* будет состоять из одной подгруппы, куда входят все пользователи, так как каждый найдёт путь до первого пользователя из своего списка.

Иначе список *subgroups* сортируется и циклически сдвигается так, что множество, содержащее U_i, становится последним в списке. Процесс повторяется, пока число подгрупп в *subgroups* не перестанет изменяться — это даст участникам полную информа-

цию о найденных связях между ними, однако метод аутентификации для такой группы неприменим.

3.2. Аутентификация пользователя

Пусть A и B связаны цепочкой групп $chain = \{G_1, \ldots, G_n\}$. Чтобы подтвердить пользователю B данные о своём открытом ключе, A формирует сообщение

$$m = (id_A, pk_A, target, chain),$$

где id_A — идентификатор A; pk_A — отпечаток его открытого ключа; $target = id_B$ — идентификатор получателя сообщения. Затем A отправляет в G_1 сообщение с групповой подписью $(m, Sig_{G_1}(m))$.

Получив такое сообщение, члены группы G_i действуют по следующему алгоритму:

1) Если $i = 1$:
   те члены группы, которые состоят в G_2, проверяют подпись группы G_1 и отпечаток открытого ключа пользователя с id_A. Если все проверки прошли успешно, то каждый из них отправляет в G_2 сообщение с цепочкой подписей $(m, Sig_{G_2}(Sig_{G_1}(m)))$.
2) Если $i \in \{2, \ldots, n-1\}$:
   те члены группы, которые состоят в G_{i+1}, проверяют подписи групп G_i, G_{i-1}, $\ldots, G_1$ и дожидаются получения сообщений от всех пользователей, принадлежащих одновременно G_{i-1} и G_i. Если все сообщения получены и информация в них совпадает, то каждый из них отправляет эти данные в G_{i+1}, предварительно дополнив цепочку подписей.
3) Если $i = n$:
   получатель, указанный в сообщении, проверяет все подписи групп $G_1, \ldots, G_n$. Так же, как и в предыдущем случае, дожидается получения сообщений от всех пользователей, принадлежащих одновременно G_{n-1} и G_n. Если все сообщения получены, информация в них совпадает и отпечаток открытого ключа отправителя сообщения равен тому, что был получен ранее, то пользователь принимает доказательство.

Надёжность такой схемы можно повысить, рассылая сообщения по множеству найденных цепочек, если таковые имеются. Тогда, приняв сообщение от A, B также находит цепочки до A. В этом случае B примет доказательство от A, если по всем установленным цепочкам групп пришло одинаковое сообщение.

3.3. Групповой вариант аутентификации

Чтобы использовать предложенный метод для аутентификации группы пользователей, воспользуемся методом Круговой аутентификации [5]. Приведём его краткое описание.

Создавая ориентированное кольцо, пользователи проходят процесс парной аутентификации. Каждый участник аутентифицирует своего соседа по часовой стрелке; если он не прошел проверку, то выбирается следующий за ним участник. В случае успешной аутентификации пара формирует группу доверенных пользователей. Процесс повторяется до первого успешного результата. При этом составляются списки успешно и неуспешно прошедших проверку. Под статусом участника понимается степень доверия к нему. Статус может принимать значения GOOD, BAD, UNKNOWN. Далее пользователи обмениваются полученными результатами. Алгоритм обработки сообщений:

— если статус отправителя UNKNOWN: сообщение хранится до момента уточнения статуса;

— если статус отправителя GOOD: пользователи со статусом UNKNOWN, состоящие в полученном списке доверенных или нарушителей, приобретают статусы GOOD и BAD соответственно и добавляются в соответствующие списки получателя;
— если статус отправителя BAD: пользователи со статусом UNKNOWN, состоящие в полученном списке доверенных участников, получают статус BAD после того, как подтвердится, что все они принадлежат одной группе доверенных пользователей.

В результате каждый из участников будет обладать информацией об аутентичности остальных членов группы. Безопасность метода обоснована стойкостью схемы цифровой подписи (в данном случае используется RSA, стойкость основана на вычислительной сложности задачи факторизации) и метода парной аутентификации. Применяя метод цепочек групп в качестве парной процедуры аутентификации, получаем групповой вариант предложенного метода аутентификации.

3.4. Исследование безопасности

Рассмотрим парную аутентификацию пользователей A и B через цепочку групп $G_1, \ldots, G_n$, где за пересылку сообщений отвечают пересечения $G_1 \cap G_2$, …, $G_i \cap G_{i+1}$, …, $G_{n-1} \cap G_n$. При этом B примет доказательство A, если до него дойдут сообщения с данными пользователя A, подписанные участниками всех этих групп, и проверки всех групповых подписей пройдут успешно.

Аутентификация методом цепочки групп предполагает пересылку сообщений с доказательствами из группы в группу. Под группой понимается множество аутентифицированных между собой пользователей, находящихся на этапе коммуникации протокола mpOTR. В этом случае внешний нарушитель, не обладающий групповым секретом и индивидуальным секретом проверенного собеседника, не сможет подменить передаваемые данные.

Перейдём к случаю, когда нарушитель — член некоторых групп из цепочки, состоящий в их пересечении. Рассмотрим несколько ситуаций, когда под контролем злоумышленника находится множество пользователей в пересечении групп:

1) Злоумышленник контролирует все узлы, находящиеся в пересечении G_i и G_{i+1}. Тогда ему необходимо состоять во всех группах цепочки, чтобы уметь переподписывать подменённое сообщение от A к B и от B к A.
2) Злоумышленник контролирует все узлы, находящиеся в пересечении G_1 и G_2. Тогда данные пользователя A могут быть подменены. Аналогично: нужно контролировать все узлы пересечения групп G_{n-1} и G_n для изменения отпечатка открытого ключа B.
3) Подконтрольные злоумышленнику узлы существуют в каждом пересечении цепочки $G_1, \ldots, G_n$, но полностью не контролируется ни одно из них. Тогда неискажённые данные достигнут честных пользователей цепочки и, получив противоречивые данные, пользователи этого пересечения не станут отправлять их дальше, будет превышено время ожидания и процесс аутентификации завершится неуспешно.

Если сообщения посылаются по множеству найденных цепочек и принимаются только в случае совпадения всех присланных данных, то злоумышленнику для подмены отпечатков открытых ключей нужно контролировать процесс передачи доказательств по всем этим цепочкам.

Однако если злоумышленник состоит в пересечении групп некоторой цепочки, он может помешать успешной аутентификации двух пользователей, если удержит валид-

ное сообщение с доказательством. В этом случае будет превышено время ожидания и процесс завершится неуспехом.

В случае атаки Сивиллы, в ходе которой жертва — пользователь A — окружён только узлами, контролируемыми злоумышленником, как описано в п. 2, B получит подменённый отпечаток ключа A. Для того чтобы убедить A принять искажённое доказательство от B, злоумышленнику необходимо состоять во всех группах всех найденных цепочек либо полностью окружать пользователя B.

Пусть злоумышленник имеет подконтрольные ему узлы в каждой аутентифицированной группе сети. Для контроля передачи данных по цепочке он должен полностью контролировать некоторое пересечение групп в ней. Поэтому рассмотрим случай, когда были созданы аутентифицированные группы из честного пользователя сети и контролируемого узла. В этом случае можно полагать, что для любых двух пользователей существует цепочка G_1, G_2, G_3, объединяющая их (рис. 1).

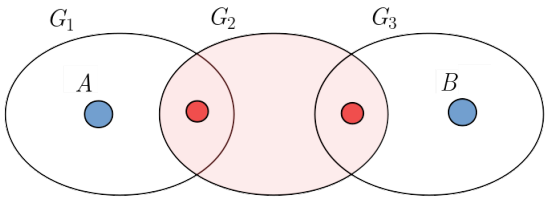


Рис. 1. Цепочка групп G_1, G_2, G_3, где красным обозначены подконтрольные злоумышленнику объекты

Этот случай подходит под описание п. 2. Если цепочка такого типа является единственной, объединяющей A и B, то атака сработает и открытые ключи будут подменены. В противном случае существует цепочка, сформированная до начала атаки, где пересечение не контролируется полностью, а значит данные, полученные по ней, не совпадут с данными первой цепочки.

Таким образом, необходимым условием успешной работы атаки является контроль некоторого пересечения групп в цепочке, причём нельзя получить полный контроль над пересечением групп, созданных до атаки, это возможно лишь при создании новых групп. Поэтому нельзя гарантировать успешную работу атаки даже в случае, когда злоумышленнику удалось войти в доверие к каждому пользователю сети.

3.5. Оценка сложности

Пусть n — число участников в группе. В табл. 1 приведены оценки сложности используемых процедур.

Таблица 1

Алгоритм	**Число раундов**	**Количество возведений в степень**	**Число сообщений**
CCEGK.Setup	$\log n$	$2\log n - 2$	$2n - 1$
CCEGK.Join	1	1	2
CCEGK.Leave	1	$3\log n - 3$	1
DGS.Sign	–	$7n - 2$	–
DGS.Verify	–	$7n$	–

Каждый пользователь в пересечении групп некоторой цепочки, число звеньев которой ограничено числом $maxNum$, выполняет $maxNum$ запусков процедуры DGS.Verify и два запуска процедуры DGS.Sign. Таким образом, число операций возведения в степень при условии, что N — среднее число участников в группах цепочки, равно

$$maxNum \cdot 7N + 2(7N - 2).$$

Участники, запустившие процесс аутентификации, находятся в концах цепочки и выполняют

$$maxNum \cdot 7N + (7N - 2)$$

операций возведения в степень.

4. Реализация алгоритма аутентификации

Алгоритм аутентификации методом цепочек групп реализован [17] и протестирован авторами данной работы. Реализация выполнена на языке JavaScript в существующем прототипе децентрализованного криптографического чата, построенного на основе P2P-топологии Chord поверх WebRTC.

Параметр $maxNum$, ограничивающий число звеньев в цепочке при её поиске, в данной реализации равен 6. В 2008 г. опубликована работа [18], в которой исследователями Microsoft проведён анализ, подтверждающий теорию о шести рукопожатиях. Основываясь на данных, полученных за месяц общения 180 миллионов пользователей мессенджера Microsoft Instant Messenger, исследователи пришли к выводу, что наиболее короткий путь, связывающий любых двух пользователей этой сети, составляет 6,6 человек. Аналогичные исследования проводились на базе социальной сети Facebook: в 2011 г. сеть состояла из 721 миллиона пользователей и размер минимальной цепочки был 4,7 [19], а в 2016 г. этот параметр стал равен 3,5, когда число пользователей достигло 1,59 миллиарда [20].

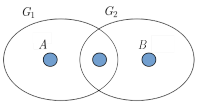
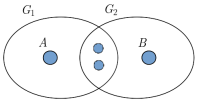
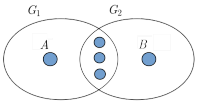
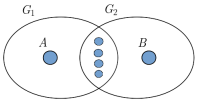
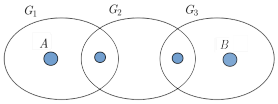
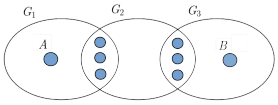
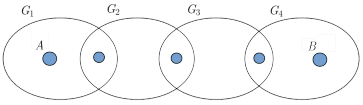
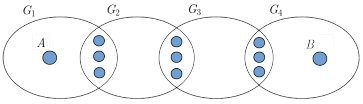
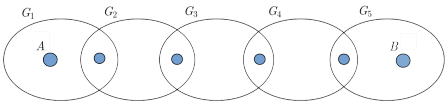
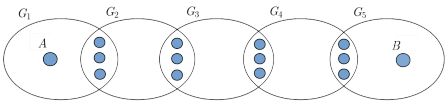
4.1. Тестирование

Тестирование проведено в браузере Chromium 81.0.4044.138. Были созданы конфигурации цепочек, описанные в табл. 2. В браузере запускались несколько экземпляров приложения — по числу пользователей, изображённых в этой таблице. Приведено время работы алгоритма аутентификации с помощью цепочек групп по уже установленной цепочке.

На рис. 2 приведён график зависимости времени работы алгоритма аутентификации от числа пользователей в пересечении групп, построенный по данным табл. 2 (конфигурации № 1–4), а на рис. 3 — график зависимости времени работы алгоритма аутентификации от числа звеньев в цепочке. График *1* соответствует конфигурациям № 1, 5, 7 и 9, где в пересечениях по одному пользователю, график *2* — конфигурациям № 3, 6, 8 и 10, где в пересечениях по три пользователя. Из рис. 2 и 3 видно, что время аутентификации возрастает практически линейно как с увеличением числа пользователей в пересечении групп, так и с увеличением числа звеньев в цепочке из групп.

Таблица 2

Время процесса парной аутентификации A и B по установленной цепочке

№ п/п	Конфигурация	Время, мс
1		2899
2		7006
3		14706
4		17330
5		12074
6		40841
7		25022
8		90752
9		30717
10		130314

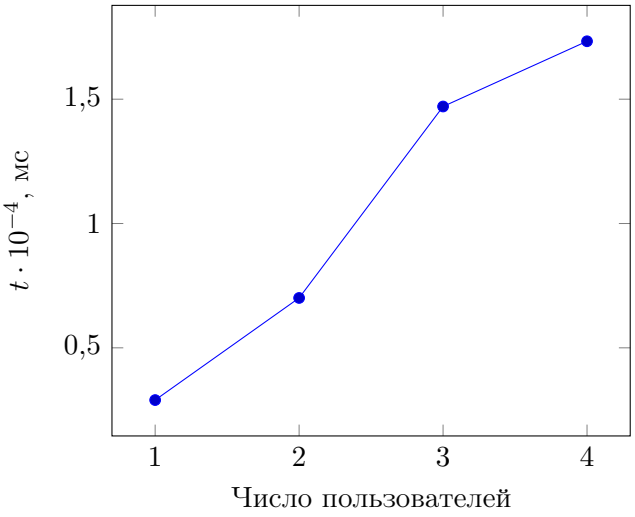


Рис. 2. Зависимость времени аутентификации t от числа пользователей в пересечении групп

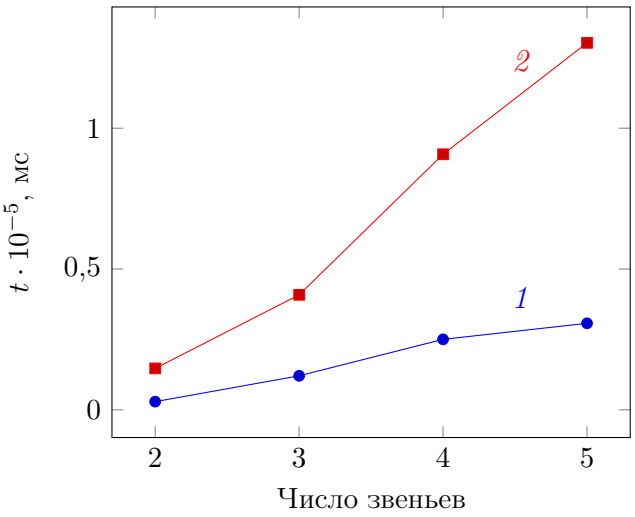


Рис. 3. Зависимость времени аутентификации t от числа звеньев в цепочке из групп пользователей

Заключение

В работе предложен алгоритм взаимной аутентификации пользователей без установления общего секрета по сторонним каналам и проведено исследование его устойчивости к атаке «человек в середине» и атаке Сивиллы.

Результаты экспериментального исследования на прототипе децентрализованной системы обмена сообщениями позволяют считать схему аутентификации применимой для сетей с развитыми социальными связями и большим количеством групп. Алгоритм взаимной аутентификации пользователей в децентрализованной системе может стать основой для создания полностью децентрализованных систем мгновенного обмена сообщениями с поддержкой свойств конфиденциальности в групповом общении, что в настоящее время не реализуется существующими системами мгновенного обмена сообщениями, прежде всего в силу наличия доверенной третьей стороны в виде провайдера сервиса.

В качестве направлений дальнейших исследований отметим вопрос безопасного хранения групповых ключей в распределённой одноранговой сети клиентов, часть

которых может быть инструментирована или модифицирована потенциальным злоумышленником. Для повышения эффективности предложенного алгоритма возможно использование механизма балансировки дерева группы, описанного в работе [13].

ЛИТЕРАТУРА

1. *Коростелева М. В., Гамаюнов Д. Ю.* Обеспечение криптографически защищенных групповых коммуникаций с функцией отказуемости // Проблемы информационной безопасности. Компьютерные системы. 2014. № 3. С. 74–79.
2. *Goldberg I. et al.* Multi-party off-the-record messaging // Proc. 16th Conf. Computer Commun. Security. ACM, 2009. P. 358–368.
3. *Шейдаев В. Ф., Гамаюнов Д. Ю.* Отказуемые групповые коммуникации в модели глобального неограниченного злоумышленника // Прикладная дискретная математика. 2018. № 40. С. 72–86.
4. `https://bitbucket.org/Enr1g/p2p_mpotr.js` — Moscow State University Seclab mpOTR.
5. *Нгуен К. К.* Доказательство с нулевым разглашением для взаимной аутентификации пользователей группового чата. Выпускная квалификационная работа. М.: МГУ, ВМК, 2018.
6. *Boudot F., Schoenmakers B., and Traore J.* A fair and efficient solution to the socialist millionaires' problem // Discr. Appl. Math. 2001. V. 111. No. 1. P. 23–36.
7. *Manulis M.* Democratic group signatures: on an example of joint ventures // Proc. 2006 ACM Symp. Inform. Comput. Commun. Security. 2006. P. 365.
8. `https://webrtc.org/` — Real-time communication for the web.
9. *Stoica I. et al.* Chord: A scalable peer-to-peer lookup service for internet applications // ACM SIGCOMM Comput. Commun. Rev. 2001. V. 31. No. 4. P. 149–160.
10. *Alves-Foss J.* An efficient secure authenticated group key exchange algorithm for large and dynamic groups // Proc. 23rd National Inform. Systems Security Conf. 2000. P. 254–266.
11. *Kim Y., Perrig A., and Tsudik G.* Communication-efficient group key agreement // IFIP Intern. Inform. Security Conf. Boston, MA: Springer, 2001. P. 229–244.
12. *Kim Y., Perrig A., and Tsudik G.* Group key agreement efficient in communication // IEEE Trans. Computers. 2004. V. 53. No. 7. P. 905–921.
13. *Zheng S., Manz D., and Alves-Foss J.* A communication-computation efficient group key algorithm for large and dynamic groups // Computer Networks. 2007. V. 51. No. 1. P. 69–93.
14. *Camenisch J. and Stadler M.* Efficient group signature schemes for large groups // Ann. Intern. Cryptology Conf. Berlin; Heidelberg: Springer, 1997. P. 410–424.
15. *Fiat A. and Shamir A.* How to prove yourself: Practical solutions to identification and signature problems // Conf. Theory Appl. Cryptogr. Techniques. Berlin; Heidelberg: Springer, 1986. P. 186–194.
16. *Boneh D., Boyen X., and Shacham H.* Short group signatures // Ann. Intern. Cryptology Conf. Berlin; Heidelberg: Springer, 2004. P. 41–55.
17. `https://github.com/naruneph/Chord_chat` — Реализация метода цепочек групп в децентрализованном чате.
18. *Leskovec J. and Horvitz E.* Planetary-scale views on an instant-messaging network // Proc. 17th Intern. Conf. World Wide Web. 2008. P. 915–924.
19. *Ugander J. et al.* The anatomy of the facebook social graph. arXiv preprint arXiv:1111.4503. 2011.
20. `https://research.fb.com/blog/2016/02/three-and-a-half-degrees-of-separation/` — Three and a half degrees of separation — Facebook Research. 2016.

REFERENCES

1. *Korosteleva M. V. and Gamayunov D. Y.* Obespecheniye kriptograficheski zashchishchennykh gruppovykh kommunikatsiy s funktsiyey otkazuyemosti [Protocol for secure group communications with deniability features]. Problemy Informatsionnoy Bezopasnosti. Komp'yuternyye Sistemy, 2014, no. 3, pp. 74–79. (in Russian)
2. *Goldberg I. et al.* Multi-party off-the-record messaging. Proc. 16th Conf. Comput. Commun. Security, ACM, 2009, pp. 358–368.
3. *Sheidaev V. F. and Gamayunov D. Y.* Otkazuemye gruppovye kommunikacii v modeli global'nogo neogranichennogo zloumyshlennika [Deniable group communications in the presence of global unlimited advisory]. Prikladnaya Diskretnaya Matematika, 2018, no. 40, pp. 72–86. (in Russian)
4. `https://bitbucket.org/Enr1g/p2p_mpotr.js` — Moscow State University Seclab mpOTR.
5. *Nguen K. K.* Dokazatel'stvo s nulevym razglasheniem dlya vzaimnoj autentifikacii pol'zovatelej gruppovogo chata [Zero-knowledge proof based authenticathion for group chat users]. Graduation Project, Moscow, MSU, 2018. (in Russian)
6. *Boudot F., Schoenmakers B., and Traore J.* A fair and efficient solution to the socialist millionaires' problem. Discr. Appl. Math., 2001, vol. 111, no. 1, pp. 23–36.
7. *Manulis M.* Democratic group signatures: on an example of joint ventures. Proc. 2006 ACM Symp. Inform. Comput. Commun. Security, 2006, p. 365.
8. `https://webrtc.org/` — Real-time communication for the web.
9. *Stoica I. et al.* Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Comput. Commun. Rev., 2001, vol. 31, no. 4, pp. 149–160.
10. *Alves-Foss J.* An efficient secure authenticated group key exchange algorithm for large and dynamic groups. Proc. 23rd National Inform. Systems Security Conf., 2000, pp. 254–266.
11. *Kim Y., Perrig A., and Tsudik G.* Communicationefficient group key agreement. IFIP Intern. Inform. Security Conf., Boston, MA, Springer, 2001, pp. 229–244.
12. *Kim Y., Perrig A., and Tsudik G.* Group key agreement efficient in communication. IEEE Trans. Computers, 2004, vol. 53, no. 7, pp. 905–921.
13. *Zheng S., Manz D., and Alves-Foss J.* A communication-computation efficient group key algorithm for large and dynamic groups. Computer Networks, 2007, vol. 51, no. 1, pp. 69–93.
14. *Camenisch J. and Stadler M.* Efficient group signature schemes for large groups. Ann. Intern. Cryptology Conf., Berlin, Heidelberg, Springer, 1997, pp. 410–424.
15. *Fiat A. and Shamir A.* How to prove yourself: Practical solutions to identification and signature problems. Conf. Theory Appl. Cryptogr. Techniques, Berlin, Heidelberg, Springer, 1986, pp. 186–194.
16. *Boneh D., Boyen X., and Shacham H.* Short group signatures. Ann. Intern. Cryptology Conf., Berlin, Heidelberg, Springer, 2004, pp. 41–55.
17. `https://github.com/naruneph/Chord_chat` — Chain of groups method realisation for decentralized chat.
18. *Leskovec J. and Horvitz E.* Planetary-scale views on an instant-messaging network. Proc. 17th Intern. Conf. World Wide Web, 2008, pp. 915–924.
19. *Ugander J. et al.* The anatomy of the facebook social graph. arXiv preprint arXiv:1111.4503, 2011.
20. `https://research.fb.com/blog/2016/02/three-and-a-half-degrees-of-separation/` — Three and a half degrees of separation — Facebook Research, 2016.