# Криптографические протоколы распределения ключей для групп с динамическим составом участников.

# Введение

В настоящее время организация безопасной связи внутри групп абонентов с динамически меняющимся составом участников является достаточно сложной задачей, отличающейся по своему качественному составу от классических задач криптографии. Она включает в себя множество сопутствующих задач, начиная от создания основных алгоритмов и заканчивая созданием конечных приложений и коммуникационных систем. Выделяют два основных аспекта безопасности при работе в группах – секретность (т. е. все взаимодействия внутри группы остаются секретными для лиц, не являющихся участниками группы) и аутентификация.

Стандартным подходом к обеспечению безопасности для групп является получение некоторой секретной величины, известной только участникам группы. Криптографические протоколы, в которых происходят выработка и распространение этой величины внутри группы известны как *распределение ключа группы (group key establishment)*. В случае, когда это значение не вырабатывается в протоколе, а приобретается заранее кем-либо из участников, протокол носит название *протокола распространения ключей в группе(group key distribution).* В случае, когда каждый участник группы участвует в генерации этого секретного значения, мы получаем *протокол* *обмена ключами (group key agreement).* В обоих случаях только действующие участники группы имеют доступ к этому групповому секрету (действующие потому, что предполагается высокая динамичность группы). При любом присоединении нового участника или выходе участника из группы секретное значение меняется для предотвращения НСД со стороны лиц, не входящих в группу.

Данная работа представляет собой обзор существующих материалов по криптографическим протоколам для динамических групп. Построена она по следующей схеме:

Раздел 1. Основные определения и понятия.

В разделе 1.1 даются основные определения.

В разделе 1.2 приводятся используемые обозначения

Раздел 2. Протоколы обмена для выработки ключа.

В разделе рассмотрены протоколы получения общего ключа для группы лиц. Приведено описание протокола Диффи-Хеллмана с аутентификацией, устойчивость его к различным атакам, протокол Диффи-Хеллмана выработки общего ключа для групп и его расширение до протокола с аутентификацией.

Раздел 3. Проект CLIQUES-API для динамических групп.

В разделе рассмотрена конкретная реализация протоколов для групп. Приведены математические основы. Тестовые величины, сравнения различных реализаций и форматы данных не рассматривались. Эту информацию можно получить из работ [3,4,5].

# 1 Основные определения и понятия

## 1.1 Основные определения

**Опр. 1.1.1.** *Протокол обмена для выработки общего ключа (key agreement protocol)* – протокол распределения ключей, в котором общий ключ вырабатывается двумя или более участниками как функция от информации, вносимой каждым из них, причем таким образом, что никакая другая сторона не может предопределить получаемый в результате общий секрет.

Протоколы обмена должны обладать следующими свойствами:

* совершенная опережающая секретность (*Perfect forward secrecy – PFS*);
* устойчивость к атакам по известному ключу (*Known-key attacks*);
* аутентификации ключа (*Key authentication*);
* подтверждение и целостность ключа (*Key confirmation & key integrity*).

Дадим некоторые определения, используемые в дальнейшем.

**Опр. 1.1.2.** Протокол обеспечивает *PFS*, если компрометация долговременных ключей не компрометирует сеансовых ключей.

**Опр. 1.1.3.** Протокол обладает свойством *контрибутивности (contributory)*, если сформированный ключ зависит от секретных данных, внесенных каждым из участников.

**Опр. 1.1.4.** Пусть *R* – протокол обмена для *n* участников, *M* – множество участников, а *Sn* – ключ, получаемый в результате протокола *R*. Тогда *R* обеспечивает *неявную аутентификацию ключа (implicit key authentication)*, если каждый *Mi*∈*M* уверен, что никакая другая сторона *Mq*∉*M* не могла получить доступ к *Sn* (за исключением злоумышленника *Mj* внутри группы).

**Опр. 1.1.5.** *Протокол аутентичного обмена для групп* – протокол обмена, в смысле опр. 1.1.4, обеспечивающий неявную аутентификацию ключа.

**Опр. 1.1.6.** Протокол обеспечивает *подтверждение ключа*, если участник протокола уверен, что другой участник (или группа) действительно обладает ключом, полученным в результате протокола.

**Опр. 1.1.7.** Контрибутивный протокол обмена обеспечивает *целостность ключа*, если участник уверен, что полученный им секретный ключ представляет собой функцию ТОЛЬКО от индивидуальных вкладов всех других участников. Таким образом, любое вмешательство в сформированный ключ (или формируемый) нарушает данное свойство, если ключ получается отличным от предполагаемого.

Список определений не полон, многие определения будут даваться по ходу рассуждений.

## 1.2 Используемые в протоколах термины и обозначения

Определим некоторые обозначения:

*n* - число участников протокола;

*i, j* - индексы для участников групп;

*Mi* - i-ый участник группы;

*G* - подгруппа *Zp\** порядка *q*, где *p* и *q* – простые;

*q -* порядок алгебраической группы;

*,g* - образующие элементы в группе *G* ;

*xi* - долговременный секретный ключ *Mi*;

*ri* - случайное (секретное) число ∈ *Zq* , вырабатываемое *Mi*;

*Sn* - групповой ключ *n* участников;

*Sn(Mi)* - вклад *Mi* -го участника в групповой ключ;

*Kij* - долговременная секретная величина, выработанная *Mi* и *Mj* , *i≠j*.

Все вычисления проводятся в циклической группе *G* простого порядка *q,* которая является подгруппой *Zp\** порядка *p*, где *p=kq+1* для некоторого k∈[[1]](#footnote-1).

Для того, чтобы предотвратить атаки методами подмены или утечки секретных величин, каждый участник протокола должен иметь возможность проверить, что те значения, которые он получает, являются элементами подгруппы.

Заметим, что *p* и *q* – общие для всех пользователей. Поскольку они вырабатываются один раз, необходимо качественно проработать процесс генерации, чтобы исключить (возможно умышленное) получение слабых или каких-то специфических простых чисел. В частности, рекомендуется использовать метод из стандарта США, описанный в FIPS 186 или же на основе метода, изложенного в стандарте ГОСТ Р34.10-94.

В таком контексте, возможности активного противника довольно сильно ограничены. Действительно, любое сообщение может быть представлено как *сmod p*, где **образующий элемент циклической подгруппы *Zp\** порядка *q* и *c* – некоторая экспонента. Получение *c* упирается в проблему дискретного логарифмирования.

# 2 Протоколы аутентичного обмена ключами

Простейшей схемой получения общего ключа является схема с доверенным сервером, в котором кто-либо посылает ему запрос на связь с другими абонентами, и сервер рассылает каждому абоненту общий ключ для связи внутри группы и список участников группы, зашифрованные ключом абонента. Но при такой схеме возникают сложности при высокой динамичности группы, обусловленные невозможностью одновременной обработки сервером большого числа запросов. Поэтому рассмотрим некоторые специально созданные протоколы для получения общего ключа участниками группы.

В рамках предварительного знакомства приведем аутентичный обмен для выработки ключа для двух сторон. Затем приведем расширение этого протокола для *n* сторон. Приводимые протоколы базируются на схеме Диффи-Хеллмана.

## 2.1 Протоколы A-DH, GDH.2 и A-GDH.2

Прежде чем привести описание протокола аутентичного обмена для двух сторон A-DH, важно подчеркнуть, что существует множество разнообразных протоколов аутентичного обмена для выработки ключа, но одни из них не поддерживают двусторонний вклад в общий ключ (как в El Gamal), другие требуют большого числа сообщений или предполагают априорный доступ к сертифицированным долговременным ключам. Многие не обладают свойством *PFS*. Поэтому, наиболее подходящим протоколом для групп в соответствии с [1] считают A-DH. Необходимо также отметить, что протокол предполагает наличие у участников аутентичных открытых ключей друг друга.

|  |
| --- |
| **Протокол A-DH.** Пусть *p,q,G –* величины, определенные выше и пусть **образующий элемент *G*.  **Предварительный этап.** Пусть *x1* и *x2* – два целых числа, т. ч. 1≤ *x1* ,*x2* ≤ *q-*1. Пусть *M1* и *M2* – два участника, которые хотят выработать общий ключ и пусть (*x1* ,*x1* mod p) и (*x2* ,*x2* mod p)- секретные и открытые ключи *M2*  и *M2* соответственно. Открытые величины системы: (*p*, *q*, ** *x1* ,*x2* ).  **Этап 1:**  *M1*выбирает случайное *r1*∈R *Zq\** ,  *M1* → *M2* : * r1* mod *p*.  **Этап 2:**  *M2* выбирает случайное *r2*∈R *Zq\** и вычисляет *K=F( x1x2* mod *p*),  *M2* → *M1* : * r2K* mod *p*.  Когда *M1* получает *J*=* r2K* mod *p*, он вычисляет *K-1* mod *q* и затем *J r1K-1* mod *p*. Получаемый в результате ключ будет *S2*=* r2r1* mod *p.* Функция *F()* может быть либо *F(x)=x* mod *q* либо *F(x)=h(x),* где *h –* хэш-функция : {0,1}\*→ *Zq\** . |

Очевидно, что в полученном в результате ключе имеется вклад обоих сторон (т.к. r1 и r2 случайны и вырабатываются разными сторонами), т.е. протокол обладает *контрибутивностью*. В то же время обеспечивается аутентификация ключа, поскольку при его формировании участвуют открытые ключи обоих абонентов, которые переданы по аутентичному каналу.

### **Теорема 2.1.1** *Протокол A-DH обеспечивает свойство PFS.*

### **Док-во:** предположим, что долговременный ключ *K=F( x1x2* mod *p*) скомпрометирован. Противник знает * r1* и * (r2K)K-1* mod *p* ≡* r2* . При данных условиях вычисление сеансового ключа *S2*=* r2r1* mod *p* эквива-лентно решению проблемы DH (Диффи-Хеллмана) для групп простого порядка. #

### Рассмотрим теперь протокол Диффи-Хеллмана для групп [2].

|  |
| --- |
| **Протокол GDH.2.** Пусть *M* = *{M1 , M2 …Mn}* – множество пользователей, которым необходимо выработать общий ключ *Sn* . GDH.2 протокол выполняется за *n* шагов. На первой стадии (*n-1* этапе) идет сбор информации от отдельных участников группы, а на второй стадии (*n* шаге) всем рассылается материал для вычисления общего ключа.  **Предварительный этап.** Пусть *p* – простое и *q –* простой делитель *p-1*. Пусть *G*-циклическая подгруппа *Zp\** порядка *q* и **образующий элемент *G.*  **Этап *i*:**  *Mi*выбирает случайное *ri*∈R *Zq\** ,  *Mi* → *Mi+1* : {* r1…ri / rj* | *j*∈[1*,i*]}, * r1…ri* .  **Этап *n*:**  *Mn* выбирает случайное *rn*∈R *Zq\** ,  *Mn* →Каждому *Mi* : {* (r1…rn) / ri* | *i*∈[1*,n*]}.  Общим ключом будет значение * r1…rn* . |

### Данный протокол можно модифицировать для обеспечения аутентификации ключа. Такая модификация отличается от выше приведенного только последним этапом. Предполагается, что *Mn* имеет с каждым *Mi* общий секрет *Kin=F(xixn* mod *p*), где *xi*-секретное долговременное значение *Mi* , *xi* mod *p* –долговременный открытый ключ *Mi* .

|  |
| --- |
| **Протокол A-GDH.2.**  **Этапы c 1 по *n-1* :** такие же, как и в GDH.2.  **Этап *n*:**  *Mn* выбирает случайное *rn*∈R *Zq\** ,  *Mn* →Каждому *Mi* : {* r1…rnKin/ ri* | *i*∈[1*,n*]}.  При получении *Mi* вычисляет * (r1…rnKin/ ri)Kin-1ri* =*r1…rn* *Sn* . |

В этом протоколе каждый участник группы вырабатывает общий аутентичный ключ с *Mn*. Более того, если мы доверяем *Mn* , то каждый участник группы может быть уверен, что такой же ключ имеют и все участники группы, т.е. они выработали общий групповой ключ. Пример протокола для четырех участников приведен на рис. 1.

Очевидно, что протокол обладает свойством контрибутивности, поскольку в результирующем ключе *Sn* есть вклад *i*-го участника группы в виде степени *ri*.

**Теорема 2.1.2** *Протокол A-GDH.2 обеспечивает свойство PFS*.

**Док-во:** Предположим, что долговременные ключи *Kin*, *i*∈[1,*n*] скомпрометированы, тогда противник в состоянии вычислить подмножество *V*={*αП(S)*| *S*⊂{*r1*,…,*rn*}}. Но, как было показано в [2], по такому *V* не возможно восстановить сеансовый ключ *Sn*. #

Рассмотрим устойчивость описанного протокола к атакам по известным ключам.

Пусть *Sn*(*Mi*) – сеансовый ключ, вычисленный каждым *Mi*. Для 0<*i*<*n*-1 можно записать его как αCiriK-1in. Для *Mn* *Sn*(*Mn*)=αCnrn, где *ci* возможно известно противнику *C*. *C* также знает подмножество {*αП(S)*| *S*⊂{*r1*,…,*rn*}}. В данных условиях нахождение α*Kin* или α*K*-1*in* невозможно, если вычислительно трудно решить проблему “распознавания Диффи-Хеллмана” [1].

Однако, некоторые из атак возможны. Если *С* попытается послать *M1* некоторое αc1 на последнем этапе протокола (где *с1* выбирается противником), то *M1* в результате получит неверный ключ Sn(*M1*)=α*c1r1K-11n*, обнаружит проблему и просто заново запустит протокол обмена. Допустим, что противник каким-то образом получил этот неверный ключ (заметим, что на практике это маловероятно). При повторном выполнении протокола *С* может подменить сообщение от *Mn-1* к *Mn* на

*αc1r1K1n-1,…,αc1r1*.

*С* подменил первое и последнее слово в сообщении, остальные слова не изменялись. Тогда *Mn* вычислит ключ *Sn*(*Mn*)=(αc1r1)rn. *Mn* также вычислит

(*αc1r1K1n-1*)*rnK1n*=*αc1r1rn*

и отошлет это значение *M1* в последнем этапе протокола. В результате *С* будет знать ключ *M1*. Но (хотя в работе [1] это и не отмечается), общий групповой ключ опять не совпадет, и через некоторое время протокол придется повторить снова. Все дело во времени определения конфликта ключей. Однако, в такого типа атаке очень много условностей, и поэтому ее можно отнести к разряду теоретических, а не к реально осуществимым атакам. Кроме того, как указано в [1], простым средством предотвращения таких атак является вычисление в качестве ключа *Sn*=*h*(*Sn*(*Mi* ), где *h()* – любая хеш-функция.

Необходимо заметить, что вышеупомянутый протокол A-GDH.2 выполняет *неявную аутентификацию ключа*, причем в довольно слабой форме, поскольку ключ не аутентифицируется непосредственно между каждыми любыми двумя *Mi* и *Mj* участниками (*i*≠*j*). Последний участник *Mn* (будем далее называть его *контролирующим группы*, поскольку через него проводятся ключевые операции протокола) отвечает за использование аутентичных ключей *Kin*, *i*=1 ... *n*-1, от которых и зависит аутентификация. Следовательно, участник *Mn* должен быть лицом, которому все доверяют. Это может быть схема с доверенным сервером в качестве *Mn*. В противном случае *Mn* сможет разбить группу на две части без обнаружения этого (поскольку он может перехватывать все сообщения и таким образом получить необходимую информацию в виде * r1…ri / rj* для ∀*i*,*j*).

Поэтому необходимо трансформировать протокол, что и было сделано в работе [1].

## 2.2 Протокол SA-GDH.2

Для описания протокола требуется несколько дополнительных определений.

**Опр. 2.2.1.** Пусть *R* – протокол обмена для *n* участников и *M* – множество участников. Мы будем говорить, что *R* является *протоколом, обеспечивающим полную(complete) аутентификацию группового ключа*, если для каждых *i*,*j* (0< *i*≠*j* ≤*n*) участники *Mi* и *Mj* вычислят общий ключ *Si,j*, *только если* *Si,j* был получен при участии каждого *Mp* ⊂ *M*.

Другими словами, полная аутентификация группового ключа есть аутентичный обмен для выработки ключа между всеми парами (*Mi*, *Mj*) при (0< *i*≠*j* ≤*n*).

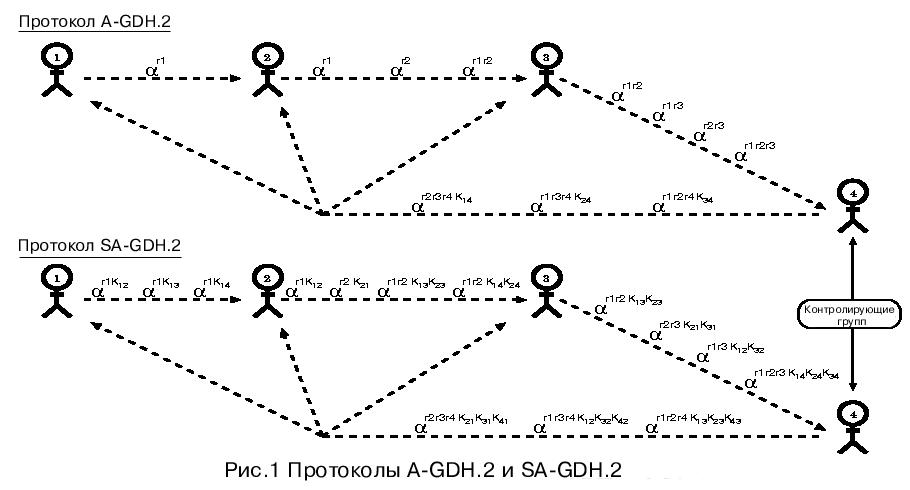
### Для выполнения этого определения можно модифицировать протокол A-GDH.2 в следующий:

|  |
| --- |
| **Протокол SA-GDH.2***.*  **Этап *i*** (0< *i* <*n)***:**   1. *Mi*получает множество промежуточных величин { *Vk* | 1≤ *k* ≤ *n*}. (в данном контексте можно сказать, что *M1* получает пустое множество на первом этапе):   * (r1…ri-1 / rk)(K k1…K k(i-1))* при *k* ≤ (*i*-1)  *Vk* =  * (r1…ri-1)(K k1…K k(i-1))* при *k* > (*i*-1) .  2. *Mi* обновляет каждое *Vk* следующим образом:  (*Vk* ) *riK ik* = * (r1…ri / rk)(K k1…K ki)* при *k* < *i*  *Vk* = (*Vk* ) *riK ik* = * (r1…ri)(K k1…K ki)* при *k* > *i .*  *Vk* при *k* = *i*  (**Замечание:** на первом этапе *M1* выбирает *V1* = **.)  **Этап *n*:**   1. *Mn* рассылает множество всех *Vk* участникам группы.   2. При получении *Mi* выбирает свое *Vi* =* (r1…rn / ri)(K1i…K ni)*.  *Mi* вычисляет (*Vk* ) *ri(K1i-1… K ni-1)*=* r1…rn* .  Заметим, что вместо того, чтобы вычислять *n* обратных элементов, *Mi* может вычислять 1 обратный элемент *Pi-1*=(*K1i-1… Kni-1*), где *Pi*=(*K1i… Kni*). |

В протоколе SA-GDH.2 требуется, чтобы каждый участник группы имел некоторый общий ключ с любым другим участником (это значение *Kji*). Однако, как уже было замечено для каждого такого ключа *Kji* не требуется нахождение обратного *Kji-1*. Таким образом, достаточно получить такие ключи перед началом работы в группе и затем эти ключи могут оставаться постоянными, не добавляя лишних действий в протокол.

Протокол основан на кольцевой схеме взаимодействия, т.е. данные проходят по кругу и лишь на последнем этапе идет широковещательная рассылка. Данные, которые передаются, представляют собой множество из *n* элементов. *i*-й элемент содержит своеобразное «накопление ключа» для *i*-го участника. Во время обновления ключа каждый из участников вносит свой вклад в формирование ключа. Только пройдя полный круг, *i*-й элемент множества будет иметь вклады всех участников и их секретные ключи для связи с *i*-м абонентом (рис. 1). Это позволяет избежать явной замены противником одного из значений на какое-то другое, выгодное ему (возможное вмешательство противника рассмотрим далее)

Однако SA-GDH.2 имеет более высокую «вычислительную стоимость» по сравнению с A-GDH.2. Прежде всего, он требует (*n*-1) экспоненцирование для каждого *Mi* (кроме первого), в отличие от *i* экспоненцирований в A-GDH.2. К этому еще добавляется работа по формированию общих ключей для каждой пары абонентов (если они не вычислены заранее): на последнем этапе проводится одно экспоненцирование – как и в A-GDH.2. На рис. 1 приведены примеры протоколов A-GDH.2 и SA-GDH.2.



Как показано в [1], протокол SA-GDH.2 обеспечивает *полную аутентификацию группового ключа*.

**Теорема 2.2.1** Протокол SA-DH обеспечивает *полную аутентификацию группового ключа*.

**Док-во:** предположим, *Mi* и *Mj* вычислили одинаковый ключ в результате выполнения протокола. Пусть *Kn*=*Sn*(*Mi*)=*Sn*(*Mj*), и предположим, что некто *Mp* ∈*M* (*p*≠*i*,*j*) не сделал вклада в этот ключ. Пусть *Vi* и *Vj* обозначают величины, полученные *Mi* и *Mj* на последнем этапе протокола. Напомним, что в соответствии с протоколом

*Sn*(*Mi*)=(*Vi*)(*K1i-1…Kni-1*)*ri* и *Sn*(*Mj*)=(*Vj*)(*K1j-1…Knj-1*)*rj* .

Поскольку все участники группы сделали вклад в ключ, то можно записать, что

*Vi*=** (*r1…rn/rpri*)(*K1i-1…Kni-1/Kpi-1*) (для *Vj* аналогично), и

*Sn*(*Mi*)= ** (*r1…rn/rp*)*Kpi-1*, что должно равняться *Sn*(*Mj*)= ** (*r1…rn/rp*)*Kpj-1.*

Но поскольку *Kpi-*1 и *Kpj-*1 являются секретными, то получаем противоречие. #

В работе [1] также отмечается, что обсуждаемый протокол обладает устойчивостью к атакам по известному ключу (*known key attacks*), однако строгого формального доказательства не приведено, авторы лишь отмечают, что это легко пронаблюдать на приведенной выше атаке на A-GDH.2.

## 2.3 Особенности ключей протоколов A-GDH.2 и SA-GDH.2

Рассмотрим важное свойство протоколов – а именно *подтверждение ключа*. Не для каждого протокола это свойство является необходимым: например, оно не нужно, если взаимодействие с полученным ключом происходит немедленно. Но тем не менее свойство подтверждения ключа является желательным для протоколов обмена по следующим причинам:

1. Протоколы обмена становятся более сильными и автономными.
2. Исключается возможность вычисления неправильного ключа без обнаружения этого (в случае перерыва между выработкой общего ключа и непосредственно передачей данных).

С другой стороны, пока еще нет точного определения, что же такое подтверждение ключа для групп. Если брать полное подтверждение ключа, то это достигается путем получения каждым участником ключа и затем доказательства всем, что он знает этот ключ.

Для протоколов A-GDH.2 и SA-GDH.2 исходя из практических соображений было определено *подтверждение ключа* как подтверждение ключа для контролирующего группы (участника *Mn*). Это может быть легко достигнуто в обоих протоколах путем добавления

***F*(*Sn*(*Mn*))

в последнее сообщение протокола (рассылка всем участникам группы), где *Sn*(*Mn*) обозначает ключ, вычисленный *Mn*, а *F* – хэш-функцию, как это было определено выше.

Таким образом, участник группы *Mi* вычисляет *Sn*(*Mi*) и проверяет равенство:

***F*(*Sn*(*Mi*)) =?***F*(*Sn*(*Mn*)) .

В [2,5] замечено, что в A-GDH.2 и SA-GDH.2 подтверждение и неявная аутентификация ключа образуют полезный в некоторых случаях побочный эффект – *аутентификацию участника Mn* для всех участников группы. Это еще раз доказывает необходимость выделения *Mn* как отдельного лица – контролирующего группы.

Включение подтверждения ключа в протокол SA-GDH.2 приводит к интересному результату, а именно:

*после выполнения протокола каждый участник группы Mi знает, что ключ, выработанный им, содержит вклад каждого участника группы.*

Это следует непосредственно из свойств *полной аутентификации группового ключа* и *подтверждения ключа*. Если бы отсутствовало подтверждение ключа, то участники группы не могли бы убедиться в истинности своего ключа.

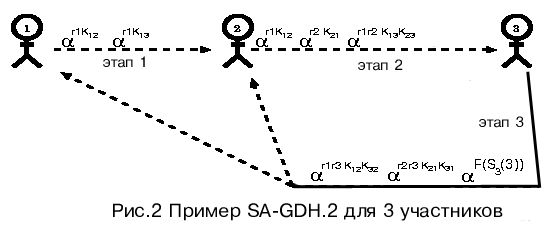
**Опр. 2.3.1.** (неформ.) Групповой протокол обмена для выработки общего ключа обеспечивает *целостность группы*, если каждый участник протокола уверен в участии каждого другого участника в протоколе.

**Опр. 2.3.2.** (неформ.) Групповой протокол обмена для выработки общего ключа является *проверяемо контрибутивным (verifiable contributory)*, если каждый участник протокола уверен, что каждый другой участник сделал вклад в групповой ключ.

Свойство проверямой контрибутивности включает в себя свойство целостности группы. Обратное утверждение неверно, поскольку целостность группы может быть обеспечена, если участник группы *Mi* будет просто подписывать сообщение отсылать его дальше. Проверяемой контрибутивности в данном случае нет. В то же время проверяемая контрибутивность может выполняться, если в формировании ключа участвовало лицо, стороннее по отношению к группе. Таким образом противник может влиять на протокол.

Однако следует заметить, что даже если выполняются свойства (неявной, полной) аутентификации и подтверждения ключа, свойство целостности ключа в вышеприведенном протоколе SA-GDH.2 не достигается.

Рассмотрим приведенный на рис. 2 пример для трех участников.



Предположим, что имеется активный противник, который может вмешиваться в передачу, подменять и модифицировать данные. Он может провести какое-либо экспоненцирование данных на этапе (1) и/или (2) и это останется незамеченным. Пусть он просто возводит в квадрат все величины на этапе (2). Тогда *M3* получит следующие значения: *α* 2*r*1*K*12, *α* 2*r*1*r*2*K*13*K*23, *α* 2*r*2*K*21.

В результате *M3* вычислит *S3*(3)= *α* 2*r*1*r*2 *r*3, т.е. квадрат предполагавшегося ключа. Все остальные участники группы получат то же самое. Подтверждение ключа здесь не помогает, поскольку злоумышленник вторгается в протокол перед тем, как *M3* вычисляет свой групповой ключ.

Как было справедливо замечено в [1], протокол SA-GDH.2 подвержен (пока) только мультипликативному (экспоненциальному) влиянию противника на ключ, т.е. можно получить ключ *Kc*, где *с*- некоторая константа, а *К*-ключ, предполагаемый в протоколе. Авторы задаются вопросом: так ли важна целостнось ключа в протоколе обмена с проверяемой контрибутивностью?

Для обеспечения целостности можно воспользоваться сторонними средствами обеспечения целостности данных во время передачи – например, проверять целостность пакетов при отправке по сети. На последнем же широковещательном этапе никаких сторонних средств не требуется, т.к любое вмешательство будет обнаружено из-за свойства подтверждения ключа для контролирующего группы.

## 2.4 Сравнение эффективности

Приведем таблицу сравнения протоколов (по вычислительным требованиям). Понятно, что различные реализации будут различаться по скорости выполнения и объему кода, а также типа используемых процессоров – это отдельная тема и с математической точки зрения не представляет интереса, нужные таблицы приведены в [2,3,4,5]. Поэтому ограничимся приведением лишь вычислительных характеристик в таблице 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Стоимость вычислений | GDH.2 | A-GDH.2 | SA-GDH.2 |
| Экспоненцирований для *Mi* | I+1 | i+1 | *n* |
| Экспоненцирований для *Mn* | *N* | *n* | *N* |
| Всего экспоненцирований | (*n*2+3*n*)/2-1 | (*n*2+3*n*)/2-1 | *n*2 |
| Вычисления обр. элементов для *Mi* |  |  | 1 |
| Вычисления обр. элементов для *Mn* |  |  | 1 |
| Всего вычислений обр. элементов |  |  | *n* |
| Умножений для *Mi* |  | 1 | 2*n*-2 |
| Умножений для *Mn* |  | *n*-1 | 2*n*-2 |
| Всего умножений |  | 2*n*-2 | 2*n*2-2*n* |

Полученные в результате приведенных протоколов общие ключи могут использоваться как ключи для секретной связи внутри группы, служить для аутентификации участников группы, выполнять роль секретного ключа при формировании групповой подписи и т.д.

# 3 Проект CLIQUES

Целью данного проекта являлась разработка протокола обмена для выработки ключа для групп. Такой протокол должен поддерживать все групповые операции по удалению, включению новых участников в группу. На основе этого протокола необходимо было создать специальный прикладной программный интерфейс (CLQ-API) , позволяющий работать приложениям в неких абстрактных группах. Протокол во многом основывается на вышеописанных протоколах аутентичного обмена. Ограничимся рассмотрением только математических принципов проекта. Не будем рассматривать полученные результаты (по эффективности), программные реализации.

В качестве базового протокола обмена для выработки общего ключа был выбран протокол A-GDH.2 (однако возможно использование и SA-GDH.2). Предполагается, что участники группы уже сформировали общий ключ.

Рассмотрим основные операции, которые позволяет выполнять разработанный протокол.

1. **Операции для одного участника группы:** включают в себя добавление или удаление одного участника группы. Данные ситуации появляются, когда кто-то хочет присоединиться к группе или покинуть ее. Эти операции могут проводится контролером группы или по согласию каждого участника группы (в зависимости от используемой политики безопасности).

2. **Операции для нескольких участников:** также включают в себя добавление и удаление. Однако есть отличия, обусловленные желанием проводить операции с несколькими участниками сразу, а не с каждым в отдельности:

* массовое присоединение: несколько участников хотят присоединиться к существующей группе;
* слияние групп: две или более групп желают соединиться в одну;
* массовый выход из группы: несколько участников хотят покинуть группу;
* разделение групп: группа распадается на две или более частей.

3. **Операции по обновлению ключа** обусловлены двумя причинами:

* ограничением шифртекста, получаемого на одном ключе для ограничения возможности получения пар открытый текст/шифртекст для проведения криптоанализа (время жизни ключа определяется выбранной политикой);
* предохранением от компрометации текущего ключа или вклада каждого участника.

Итак, список операций, выполняемых протоколом, выглядит следующим образом:

* присоединение (JOIN): новый участник добавляется в группу;
* слияние (MERGE): один или более участников добавляются в группу;
* выход из группы (LEAVE): один или более участников покидают группу;
* обновление ключа (KEY REFRESH): генерация нового ключа для группы.

Для простоты, считается, что последний участник группы является контролирующим группы (это может быть легко исправлено и не является критическим требованием).

**Присоединение**

Операция добавляет нового участника *Mn+1* к группе из *n* участников. Во время операции вычисляется новый групповой ключ *Sn+1* , и *Mn+1* становится новым контролирующим группы. Предполагая, что *Mn* является текущим контролирующим группы, протокол выглядит следующим образом:

1. *Mn* вырабатывает новое значение *rn’* и получает множество[[2]](#footnote-2) чисел

*M*={*g r1…rn’/ri* | *i*∈[1,*n*-1]} ∪{ *g r1…rn-1* }∪{ *g r1…rn’* }

Затем *M* посылается *Mn+1*.

1. После получения сообщения *Mn+1* вырабатывает число *rn+1* и вычисляет значение *g Ki,n+1r1…rn’rn+1/ri* для всех *i* из [1,*n*]. Затем это множество рассылается всей группе.
2. При получении каждый *Mi* вычисляет групповой ключ как

(*g Ki,n+1r1…rn’rn+1/ri*)K-1i,n+1ri= *g r1…rn’rn+1*= *Sn+1*. А *Mn+1* вычисляет ключ, используя сообщение из шага (1).

Шаги (1) и (2) требуют *n* экспоненцирований, шаг (3) требует одно экспоненцирование для каждого участника группы. Общее число экспоненцирований для получения ключа равно 2*n*+1 (считается, что на третьем шаге экспоненцирования происходят одновременно и по времени равны одному).

**Слияние**

Операция используется для добавления *k*>0 участников к существующей группе из *n*>1 участников. Пусть *m*=*n*+*k*. Во время операции вырабатывается новый групповой ключ *Sm*, и *Mm* становится новым контролирующим группы. Предполагая, что *Mn* является текущим контролирующим группы, протокол выглядит следующим образом:

1. *Mn* вырабатывает новое значение *rn’* и вычисляет[[3]](#footnote-3) *g r1…rn-1rn’*. Затем это сообщение отправляется к *Mn+1*.
2. Каждый участник *Mj* , *j*=*n*+1,…,*m*-1 вырабатывает число *rj* и вычисляет *gr1….rn’…rj* . Это сообщение посылается *Mj+1*.
3. После получения сообщения, *Mm* рассылает полученное значение всей группе
4. После получения сообщения каждый участник *Mi*, *i*=1,2,…,*m*-1 группы вычисляет *g(r1….rn’…rm-1)/ri* и посылает его *Mm*.
5. *Mm* вырабатывает *rm* и получает множество

*M*={ *g Ki,m r1…rn’ … rm/ri* | *i*∈[1,*m*-1]}.

Затем оно посылается группе.

1. При получении сообщения шага (5) каждый *Mi* , *i*=1,2…*m*-1 вычисляет групповой ключ как (*gr1…rn’…rmKim / ri*)*Kim*-1*ri*= *g r1…rn’…rm*= *Sm*. Аналогично, *Mm* вычисляет ключ, используя сообщение из шага (3).

Если *k*=2, то шаг (2) не нужен, в остальном протокол выглядит также.

Шаги требуют всего *k* модульных экспоненцирований. Также, как и ранее, шаги (4) и (6) требуют по одному для каждого участника. Шаг (5) требует *n*+*k*-1 экспоненцирований. Число экспоненцирований для присоединения *k* участников равно *n*+2*k*+1.

Операция присоединения также может быть использована для добавления *k* участников к группе. Это потребует повторить операцию присоединения *k* раз – соответственно возрастает трудоемкость операции. Таким образом для массового добавления участников группы лучше использовать операцию слияния. Если использовать операцию слияния для добавления одного участника к группе, то получается на два экспоненцирования больше, чем для операции присоединения. Итак, присоединение используется для добавления одного участника к группе, а слияние – нескольких.

**Выход из группы**

Операция выхода из группы удаляет *k* участников из *n* участников текущей группы. Во время операции вычисляется новый групповой ключ *Sn-k*. *Mn-k* становится новым контролирующим группы, если *Mn* покидает группу. Для простоты предположим, что только один участник *Md* выходит из состава группы. Протокол выглядит следующим образом:

1. *Mn* вырабатывает новое *rn’* и получает множество

*M*={ *g Kin r1…rn’/ ri* | *i*∈[1,*n*-1] и *i*≠*d*}

Затем *М* рассылается всем.

1. При получении сообщения *Mi* вычисляет

(*g Kinr1…rn’/ri*)*Kin*-1*ri*= *g r1…rn’*= *Sn*. *Mn* вычисляет новый групповой ключ *gr1…rn’*=*Sn* используя старое значение.

Участник *Md* не может вычислить новый групповой ключ, т.к. контролирующий группы не вычислил вспомогательный ключ *gKdnr1…rn’/rd* для *Md*. Если несколько участников покидают группу, то контролирующий группы не вычисляет нужные значения для выходящих из группы участников на шаге (1).

Если из группы выходит контролирующий, то вышеописанные операции выполняет предпоследний участник группы *Mn-1*. Более того, поскольку новый контролирующий группы не может удалить из ключа долговременные ключи (они были у прошлого контролирующего группы), каждый участник *Mi* должен заново вычислить свой случайный сеансовый ключ как *ri*= *ri*\*(*Kin-1* *mod* *q*) перед выполнением шага (2).

Шаг (1) требует *n*-*k* экспоненцирований. Шаг (2) требует одно экспоненцирование от каждого участника группы. Таким образом, операция выхода из группы требует *n*-*k*+1 модульных экспоненцирований.

**Обновление ключа**

Операция обновления ключа выполняет замену группового ключа на новый. Использование этой операции зависит от используемой политики приложения, использующего CLIQUES или политики работы с ключами для предприятия. Эта операция выглядит также, как и операция выхода из группы с *k*=0, т.е. на первом шаге *Mn* вырабатывает множество

*M*={ *g Kin r1…rn’/ ri* | *i*∈[1,*n*-1]}

для всех участников группы.

Приведенный протокол является протоколом аутентичного обмена и обладает свойством контрибутивности, что гарантирует независимость ключа (так как в его формировании участвуют все участники группы), может обеспечивать подтверждение ключа (как это было описано выше), обладает свойством *PFS* и устойчив к атакам по известному ключу (многие свойства следуют из рассмотренного ранее протокола A-GDH.2).

Таким образом, с использованием приведенных выше операций достигается полноценная работы группы. На основе приведенного протокола был разработан интерфейс прикладного программирования (*Application Programming Interface - API*). В работах [3,5] приводятся форматы заголовков сообщений и принципы построения приложений на основе CLIQUES-API. Он принят как проект стандарта для Internet. Программные реализации математических принципов, используемых в протоколах, можно найти в крипто-библиотеках Crypto++[6] и RSAREF[7].

Между тем, приведенная схема работы не лишена недостатков. Во-первых – и это, наверное, самый главный из них – приходится менять ключ для всей группы при изменении ее состояния. Это может подходить для небольших групп, но при большом числе участников становится серьезной трудностью. В этом случае все будет зависеть от динамики группы. На решение этой задачи были направлены усилия разработчиков. Также решающую роль играет пропускная способность каналов связи между участниками, поскольку в случае появления участника со слабым каналом (тем более, если это контролирующий группы) встает вопрос о временных факторах формирования ключа. Возможна ситуация непроизвольного «выкидывания» (в случае отсутствия необходимых механизмов) участника, использующего канал с низкой пропускной способностью в случае высокой динамики группы. Он просто не будет успевать получать новые данные для формирования ключа.

Во-вторых – довольно высокое число экспоненцирований в операциях протокола.

# Заключение

Рассмотренные протоколы обмена для выработки общего ключа предоставляют большие возможности для развития безопасных протоколов коммуникаций для динамических групп. На основе них можно строить сложные протоколы аутентификации, цифровой подписи, доказательства знания.

Вкратце опишем, что не вошло в данную работу.

1. Цифровые подписи – данный раздел довольно велик по своему объему.

Существует большое число проблем, связанных с подписями для групп. Прежде всего это проблемы устойчивости к атакам объединенных пользователей и проблемы удаления участников группы и их ключей. В сфере предлагаемой анонимности для участников группы это является серьезной проблемой. Также до конца не определено межгрупповые взаимодействия и случаи с одновременным членством в нескольких группах (использование нескольких ключей признается нерациональными) и образование подгрупп. Существующие схемы имеют предварительный характер.

1. Взаимодействие протоколов с сетевыми технологиями. Ввиду сугубо практических аспектов данные материалы не учитывались

В настоящее время задачам безопасной связи внутри групп с динамическим составом участников уделяется повышенное внимание благодаря повсеместному использованию технологий сети Internet. Возможно, в скором времени появятся новые протоколы распределения ключей, не содержащие недостатков описанных протоколов.

# Используемые работы:

[1] G. Ateniese, M. Steiner, G. Tsudik “Authenticated Group Key Agreement and Friends”, in *ACM Symposium on Computer and Communication Security*, November 1998.

[2] M. Steiner, G. Tsudik, M. Waidner “Diffie-Hellman key distribution extended to groups”, in *ACM Conference on Computer and Communications Security*, pp.31-37, ACM Press, Mar. 1996.

[3] G. Ateniese, D. Hasse, O. Chevassut, Y. Kim, G. Tsudik “The Design of a Group Key Agreement API”, IBM Research Division, Zurich Research Laboratiry.

[4] Y. Amir, G. Ateniese, D. Hasse, Y. Kim, C. Nita-Rotaru, T. Sclossnagle, J. Schultz, J. Stanton, G. Tsudik “Secure Group Communications in Asynchronous Networks with Failures: Integration and Experiments”, 1999.

[5] G. Caronni, M. Waldvoget, D. Sun, B. Plattner “Efficient Security for Large and Dynamic Multicast Groups”, Computer Engineering and Networks Laboratory.

[6] W. Dai “Crypto++”, 05.1999, http://www.eskimo.com/~wedai/cryptolib.html

[7] RSA Laboratories, http://www.rsalab.com

1. может быть вычислен посредством выбора случайного элемента *b∈Zp\** и вычисления *b(p-1)/q mod p* до тех пор, пока ≠ [↑](#footnote-ref-1)
2. Необходимые данные для вычисления множества *Mn* берет из последнего этапа протокола A-GDH.2 и возводя затем нужные элементы в степень *rn’(Kin*-1*mod p)* получает необходимые значения. [↑](#footnote-ref-2)
3. Значение *g r1…rn-1* *Mn* может получить из предыдущего ключа путем возведения в степень

   *rn-1* . [↑](#footnote-ref-3)