УДК 004.056.5

КРИПТОГРАФИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ КЛЮЧЕЙ ДЛЯ ЗАЩИТЫ ИНФОРМАЦИИ В ИЕРАРХИЧЕСКИХ СИСТЕМАХ

***В. Д. Лернер*[1],**
начальник отдела информационно-коммуникационного обеспечения
ООО «Космос СПб», г. Санкт-Петербург

Рассматриваются современные системы многоуровневой защиты информации, приводятся ключевые достоинства систем и обосновываются их недостатки. На основе формируемых требований к таким системам предлагается комбинированный алгоритм для криптографического распределения ключей.

Ключевые слова *— защита информации, иерархические системы, криптографическое распределение ключей.*

Введение

Многоуровневая защита обеспечивает разграничение доступа субъектов с различными правами доступа к объектам различных уровней конфиденциальности [1].

Системы многоуровневой защиты предназначены для того, чтобы пользователи имели доступ только к тем частям автоматизированной системы обработки информации, на которые они имеют полномочия, и не могли получить доступ к другим частям автоматизированной системы.

Существуют различные формы разграничения доступа к информации в многоуровневых системах, но наиболее часто встречается модель, построенная по иерархическому принципу [2].

Иерархический доступ

Иерархическую модель графически можно представить в виде однонаправленного родового дерева [3]. Корень может иметь произвольное число подчиненных (порожденных) элементов, у каждого из которых может быть произвольное число подчиненных элементов более низкого уровня, и так далее для любого числа уровней. У такого дерева корень — администратор системы, имеющий доступ ко всей информации в системе; узлы — пользователи различных уровней иерархии, имеющие доступ к своей информации и информации подчиненных им пользователей; листья — пользователи, имеющие доступ только к своей информации. Ветви дерева являются однонаправленными сверху вниз и показывают возможные пути доступа к информации в системе. Принципиальным для иерархии является то, что каждый элемент приобретает свой смысл только тогда, когда он рассматривается в своем контексте, т. е. подчиненный элемент не может существовать без своего предшественника по иерархии.

Существует три основных метода построения систем защиты информации с иерархическим доступом:

1) административное распределение ключей;
2) иерархическое шифрование;
3) криптографическое распределение ключей.

Подробный анализ вышеперечисленных методов приведен в работах [2, 4], здесь же следует отметить, что метод криптографического распределения ключей, которому посвящена предлагаемая работа, позволяет использовать для шифрования информации стандартные широко известные алгоритмы, например ГОСТ 28147-89, DES, AES.

Впервые постановка и решение задачи криптографического распределения ключей в иерархических системах была предложена Аклом (Akl) и Тэйлором (Taylor) в 1983 г. [5].

В качестве иерархической системы Akl и Taylor рассматривали информационную, коммуникационную систему с общим числом пользователей $|S|$, в которой пользователи U_i, $i \in S$ обладают разными полномочиями. Пользователи частично упорядочены отношением $\subseteq$, где $U_i \subseteq U_j$ означает, что пользователь U_j может иметь доступ к информа-

[1] Научный руководитель — доктор технических наук, профессор, доцент кафедры комплексной защиты информации Санкт-Петербургского государственного университета аэрокосмического приборостроения *С. В. Беззатеев*.

ции своего подчиненного U_i; если обозначить операцию доступа символом R, то U_jRU_i. По определению, все пользователи системы подчинены администратору системы — пользователю U_0, т. е. $U_i \subseteq U_0$, $\Rightarrow U_0RU_i$, $\forall i \in S\backslash\{0\}$. Администратор системы генерирует и раздает ключи всех пользователей системы K_i, $i \in S\backslash\{0\}$, а также выбирает симметричный алгоритм шифрования, который является единым алгоритмом шифрования для всех пользователей системы. Таким образом, каждый пользователь системы U_i шифрует собственную информацию, используя единый для всей системы алгоритм шифрования, выбранный администратором, и свой секретный ключ K_i.

Задача иерархического доступа к информации в такой системе сводится к генерированию таких ключей пользователей, которые бы позволяли осуществлять доступ к информации согласно полномочиям пользователей. Очевидно, что администратор, зная ключи всех пользователей системы, может дешифрировать по известному алгоритму шифрования информацию любого пользователя системы, т. е. U_0RU_i, $i \in S\backslash\{0\}$.

Общий вид иерархической системы показан на рис. 1.

Введем следующие понятия, которые присущи родовому дереву [3]:

1) под отцами и сыновьями понимаются пользователи, которые стоят на соседних уровнях, и отец имеет доступ к информации сына;

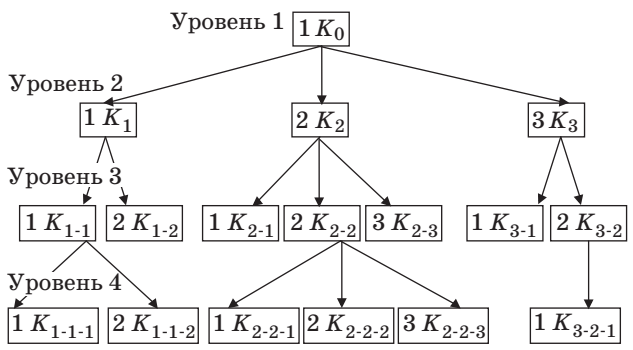
2) под прародителями потомка U_i понимаются такие пользователи U_j, что U_jRU_i;

3) пользователи, не имеющие сыновей, называются конечными пользователями;

4) под братьями понимаются пользователи, имеющие одного отца.

Каждому пользователю системы можно присвоить свой уникальный номер — идентификатор, который состоит из номеров прародителей всех предшествующих уровней (кроме 1-го) и своего номера (см. рис. 1).

Основные требования, предъявляемые к данному классу задач:



■ ***Рис. 1***. Общий вид иерархической системы

1) пользователь системы должен иметь доступ к информации только согласно своим полномочиям;

2) коалиция пользователей не должна иметь доступ к информации не подчиненных им пользователей;

3) вычисление ключей пользователей на любом уровне не должно приводить к замедлению работы автоматизированной системы обработки информации;

4) система должна обеспечивать возможность увеличения как в глубину, так и в ширину, без перерасчета остальных параметров системы, т. е. при увеличении количества пользователей на любом уровне, кроме первого, не должен происходить пересчет ключей пользователей на остальных уровнях.

В данной статье не будет рассмотрен класс алгоритмов, которые для получения ключей используют либо выделенный сервер [6], либо смарт-карты [7].

Все существующие криптографические алгоритмы распределения ключей можно разделить на два класса:

1) алгоритмы «снизу-вверх», в которых либо открытый ключ, либо секретный ключ администратора содержит в себе ключи всех пользователей, начиная с самых нижних уровней;

2) алгоритмы «сверху-вниз», в которых ключ администратора не содержит в себе ключей потомков, но позволяет вычислить их по некоторому алгоритму.

Алгоритм Дж. Иеха и Р. Шьяма

Одним из последних алгоритмов класса «снизу-вверх», родоначальниками которого были Akl и Taylor [5], является алгоритм Дж. Иеха (J. Yeh) и Р. Шьяма (R. Shyam) [8].

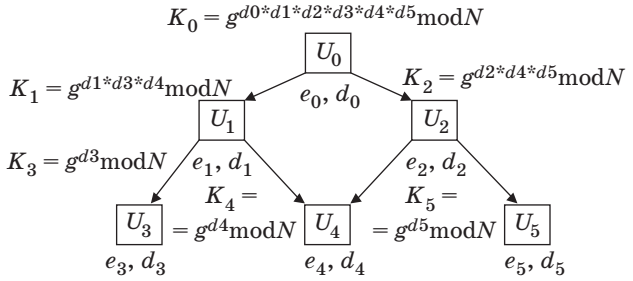
Схема алгоритма заключается в следующем.

Администратор выбирает два больших простых секретных числа p и q и вычисляет $N = p*q$ и $\varphi(N) = (p-1)*(q-1)$. Для каждого пользователя U_i выбирает простое число e_{Ui}, взаимнопростое с $\varphi(N)$, затем вычисляет d_{Ui} такое, что $e_{Ui}*d_{Ui} \equiv 1 \bmod \varphi(N)$. Также администратор выбирает секретное число g, являющееся основанием по модулю N, а затем публикует только e_{U1}, ..., e_{Us} и N и генерирует ключи для всех пользователей по следующей формуле:

$$K_j = g^{\Pi_{Ui \subseteq Uj} d_{Uj}} \bmod N,$$

т. е. аналогично всем подобным схемам в ключ отца U входят ключи d_{Ui} его потомков $U_i \subseteq U_j$ и его собственный ключ d_{Uj}.

Если пользователь U_i хочет вычислить ключ пользователя U_t, он применяет следующую формулу:



■ ***Рис. 2.*** Пример работы алгоритма J. Yeh и R. Shyam

$$K_t = K_i^{\prod_{Uj \neq Ut} e_{Uj}} \mathrm{mod} N,$$

т. е. возводит свой ключ в степень произведения своего открытого ключа и открытых ключей всех своих потомков, кроме открытого ключа e_t пользователя U_t. Справедливость и безопасность алгоритма доказана в работе [8].

Пример работы алгоритма для шести пользователей приведен на рис. 2.

Для вычисления ключа K_4 пользователь K_1 выполняет следующую операцию: $K_4 \equiv K_1^{e1*e3} \mathrm{mod} N \equiv \equiv g^{(d1*d3*d4)*e1*e3} \mathrm{mod} N \equiv g^{d4} \mathrm{mod} N$.

Существенным недостатком данного метода распределения ключей является необходимость переконфигурировать различные части системы при добавлении хотя бы одного пользователя U_i. В этом случае следует создать новые d_{Ui}, и e_{Ui} и заново пересчитать все ключи прародителей пользователя U_i. Также при удалении пользователя U_i следует заново рассчитать все ключи прародителей пользователя U_i, что приводит к необходимости дешифрировать всю ранее зашифрованную этими пользователями информацию и шифровать ее с новыми ключами, что является очень трудоемкой и небезопасной операцией.

Алгоритм Сандху

Второй подход к решению задач распределения ключей «сверху-вниз» в 1988 г. предложил Сандху (Sandhu) [9]. Идея алгоритма основывается на однонаправленных функциях, т. е. функциях, значения которых легко получить в одну сторону, но невозможно в обратную.

Стойкая хэш-функция должна обладать следующими свойствами:

1) односторонностью: пусть дано хэш-значение H(M) некоторого неизвестного сообщения M. Тогда вычислительно невозможно определить M по имеющемуся H(M);

2) стойкостью к коллизиям: пусть дано сообщение M и его хэш-значение H(M). Тогда вычислительно невозможно (вычислительно сложно) подобрать $M' \neq M$ такое, что H(M) = H(M');

3) строгой стойкостью к коллизии: вычислительно невозможно найти два произвольных различных сообщения M и M', для которых H(M) = = H(M').

Для лобовой атаки на однонаправленные хэш-функции используют два метода. Первый направлен на взлом второго свойства, т. е. по значению хэш-функции H(M) противник хочет создать другой документ M', такой, что H(M') = H(M). Другой метод направлен на взлом третьего свойства: противник хочет найти два случайных сообщения M и M', таких, что H(M) = H(M').

Подробный анализ взлома хэш-функций приведен в работе [10], здесь же оценим, насколько успешными на практике могут быть атаки, основанные на двух описанных выше методах. Пусть одна MIPS (Million Instruction Per Second) машина хэширует миллион сообщений в секунду. При таких условиях число хэш-значений, вычисленных одной MIPS-машиной за один год, составляет $L = 3{,}15*10^{13}$. Оценка вероятности взлома хэш-функции для двух рассмотренных методов атаки при различных значениях длины выходного хэш-значения приведена в табл. 1.

Из таблицы видно, что при одинаковой длине значения хэш-функции вероятность взлома первым методом намного ниже, чем при взломе хэш-функции вторым методом.

Таким образом, чтобы обеспечить требуемую вероятность устойчивости хэш-функции к взлому, необходимо использовать большую длину значения хэш-функции. Так, например, при требовании обеспечить вероятность взлома не более 10^{-30}, необходимо использовать не 128-битное, а 256-битное значение хэш-функции.

Также можно сделать вывод, что при одинаковой длине значения хэш-функции на ее взлом методом поиска документа $M' \neq M$, такого, что H(M') = H(M), потребуется гораздо меньше времени, чем при взломе хэш-функции методом, основанном на парадоксе «дней рождений» [10].

Таким образом, чтобы обезопасить хэш-функцию от взлома на заданный интервал времени,

■ ***Таблица 1***

Длина хэш-значения, бит	Первый метод		Второй метод	
	Вероятность взлома	Продолжительность взлома, MIPS-лет	Вероятность взлома	Продолжительность взлома, MIPS-лет
64	$1{,}08 \times 10^{-19}$	300 000	$2{,}33 \times 10^{-10}$	1,19 ч
128	$5{,}88 \times 10^{-39}$	$5{,}4 \times 1024$	$5{,}42 \times 10^{-20}$	600 000
256	$1{,}73 \times 10^{-77}$	$1{,}8 \times 1063$	$2{,}94 \times 10^{-39}$	$1{,}1 \times 1025$
512	$1{,}49 \times 10^{-154}$	2,1 × 10 140	$8{,}64 \times 10^{-78}$	$3{,}7 \times 1063$

необходимо использовать большую длину значения хэш-функции. Так, при требовании обеспечить стойкость к взлому хэш-функции в течение 1,1 × 1025 MIPS-лет необходимо использовать не 128-битное, а 256-битное значение хэш-функции.

Российский стандарт хэш-функции ГОСТ Р 34.11-94 использует 256-битное значение хэш-функции, что позволяет утверждать, что при современных вычислительных мощностях его компрометация вычислительно невозможна. Другими словами, для взлома хэш-функции ГОСТ Р 34.11-94 вторым, более эффективным методом потребуется 1,1 × 1025 MIPS-лет.

Алгоритм Хвона

В 1999 г. Хвонг (Hwang) предложил схему распределения ключей для частично-упорядоченной иерархии. Алгоритм состоит из двух частей: вычисления секретного ключа на основе ключей потомков и получения из секретного ключа открытого ключа при помощи симметричного алгоритма шифрования (вместо хэш-функции) и своего секретного ключа. Процедура генерирования ключей следующая:

1) администратор каждому пользователю присваивает уникальный секретный ключ K_i такой, что если $U_j R U_i$, то $K_j > K_i$, и безопасно распространяет их между пользователями;

2) администратор выбирает симметричный алгоритм шифрования $Ek_i(x)$, делая его открытым;

3) администратор вычисляет открытые ключи пользователей PD_i, у которых есть потомки K_{ij}, по следующему алгоритму:

$$SD_i = \sum_1^r K_{ij} K_i^{j-1},$$

где r — количество потомков пользователя U_i;

$$PD_i = Ek_i(SD_i).$$

Для вычисления ключей потомков пользователь должен выполнить следующую операцию по уровням дерева, пока не дойдет до требуемого уровня:

1) получить из открытого ключа закрытый ключ $SD_i = Dk_i(PD_i)$;

2) вычислить K_{ij} по формуле

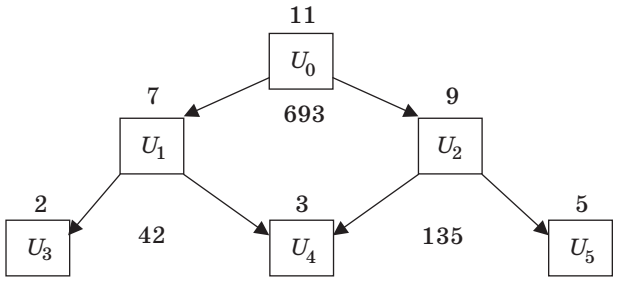
$$K_{ij} = \left\lfloor SD_i / K_i^{j-1} \right\rfloor \bmod K_i,$$

где $\lfloor\ \rfloor$ — округление вниз до целого. Справедливость формулы доказана [11].

Пример работы алгоритма для шести пользователей приведен на рис. 3, где над пользователями указаны K_i, а под ними — SD_i.

Можно отметить следующие недостатки данного метода:

1) уменьшение размеров секретных ключей в зависимости от уровня иерархии;



■ ***Рис. 3.*** Пример работы алгоритма Hwang

2) большие размеры открытых ключей пользователей, т. е. необходимость проводить вычисления с очень большими числами;

3) для получения ключа пользователя на 5 уровней ниже необходимо 5 раз выполнить симметричный алгоритм дешифрирования, что снижает производительность системы по сравнению с предыдущими алгоритмами.

К достоинствам данного метода следует отнести малые размеры секретных ключей, возможность изменения структуры иерархии без переконфигурирования ключей системы, а также применимость во всех иерархических структурах.

Комбинированный алгоритм

До сих пор предлагаемые системы криптографического распределения ключей при одинаковой криптостойкости алгоритмов удовлетворяли либо третьему, либо четвертому требованию, предъявляемому к данному классу алгоритмов (см. выше). Поэтому комбинированный метод распределения ключей, позволяющий эффективно вычислять ключи пользователей, а также модифицировать систему, является актуальной задачей для систем, построенных по иерархическому принципу.

Для решения этой задачи в представленном комбинированном алгоритме [12] используется неопределенность, возникающая при целочисленном делении [13], которая определяется задачей поиска подходящего остатка при известном делителе и частном. Очевидно, что объем перебора значительно снижается при возможности коалиционной атаки и при этом существенно зависит от числа участников коалиции.

Схема генерирования и распределения ключей

1. Администратор системы выбирает случайное большое секретное число K_0. По мере возникновения 2-го уровня иерархического дерева администратор для пользователей $U_1, U_2, \ldots, U_Q$ генерирует Q открытых ключей $T_1, T_2, \ldots, T_Q$, представляющих собой числа одного порядка, мень-

шего, чем K_0, и вычисляет Q секретных ключей $K_1, K_2, ..., K_Q$ по следующему алгоритму:

а) администратор целочисленно делит секретный ключ K_0 на открытые ключи $T_1, T_2, ..., T_Q$ и вычисляет Q частных $S_1, S_2, ..., S_Q$ и Q остатков $R_1, R_2,..., R_Q$:

$$S_1 = K_0 \text{div} T_1,\ S_2 = K_0 \text{div} T_2,\ ...,\ S_Q = K_0\ \text{div} T_Q,$$

где div — операция деления без остатка;

$$R_1 = K_0 - S_1{*}T_1,$$
$$R_2 = K_0 - S_2{*}T_2,\ ...,\ R_Q = K_0 - S_Q{*}T_Q;$$

б) администратор вычисляет ключи для пользователей 2-го уровня:

$$K_1 = S_1{*}R_1,\ K_2 = S_2{*}R_2,\ ...,\ K_Q = S_Q{*}R_Q.$$

2. Администратор для братьев вычисляет ключ группы $K_{0\text{гр}}$, для чего генерирует Q открытых ключей группы $T_{10}, T_{20}, ..., T_{Q0}$ так, чтобы выполнялось условие $K_{0\text{гр}} = K_1 \text{div} T_{10} = K_2 \text{div} T_{20} = ... = K_Q \text{div} T_{Q0}$.

3. По мере возникновения 3-го уровня иерархического дерева администратор для пользователей 3-го уровня выполняет операции по п. 1 и 2, где вместо своего секретного ключа K_0 использует секретные ключи отцов пользователей 3-го уровня (ключи $K_1, K_2, ..., K_Q$). Аналогичные операции администратор выполняет при возникновении остальных уровней системы.

4. При добавлении нового подчиненного U_y на любой уровень, кроме первого, администратор выполняет следующие операции:

а) вычисляет K_x — секретный ключ пользователя U_x, отца пользователя U_y:

$$K_x = ((((K_0 \text{div} T_{xn}{*}R_{xn}) \text{div} T_{xn-1}{*}R_{xn-1})...$$
$$\text{div}\ T_{x1}{*}R_{x1})) \text{div}\ T_x{*}R_x),$$

где $T_{xn} ... T_{x1}$ — открытые ключи потомков пользователя U_0 по иерархическому дереву до отца пользователя U_x; $R_{xn} ... R_{x1}$ — остатки, получаемые при целочисленном делении на соответствующие открытые ключи; T_x — открытый ключ пользователя U_x; R_x — остаток, получаемый при целочисленном делении согласно вышеприведенной формуле на открытый ключ T_x;

б) генерирует T_y — открытый ключ пользователя U_y;

в) делит секретный ключ K_x, вычисленный по п. 4а, на открытый ключ T_y и вычисляет частное S_y и остаток R_y: $S_y = K_x \text{ div } T_y$, $R_y = K_x - S_y{*}T_y$;

г) вычисляет ключ K_y: $K_y = S_y{*}R_y$.

Каждому пользователю системы можно присвоить свой уникальный номер — идентификатор, который состоит из номеров прародителей всех предшествующих уровней (кроме 1-го) и своего номера. Например, для схемы, изображенной на рис. 4, распределение идентификаторов представлено в табл. 2, где символ «-» означает разделение номеров.

Пример работы алгоритма по распределению ключей для иерархической системы из 11 пользователей (см. рис. 4) приведен в табл. 3.

При добавлении пользователя $U_{1\text{-}3}$ администратор формирует для него следующие значения согласно п. 4:

а) вычисляет ключ пользователя U_1:

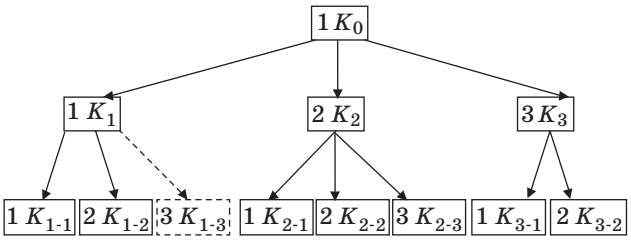
$$K_1 = (K_0 \text{div} T_1){*}R_1 =$$
$$= (987\ 654\ 321 \text{ div } 4234){*}1843 = 429\ 911\ 081;$$

б) генерирует:

$$T_{1\text{-}3} = 8237;$$

в) вычисляет:

$$S_{1\text{-}3} = K_1 \text{div} T_{1\text{-}3} = 429\ 911\ 081 \text{div } 8237 = 52\ 192,$$



■ ***Рис. 4.*** Пример иерархической системы из 11 пользователей

■ ***Таблица 2***

Уровень	Идентификаторы						
1	0						
2	1		2			3	
3	1-1	1-2	2-1	2-2	2-3	3-1	3-2

■ ***Таблица 3***

Пользователь	T	S	R	K	$T_{\text{гр}}$	$R_{\text{гр}}$	$K_{\text{гр}}$
U_0	–	–	–	987 654 321	–	–	–
U_1	4234	233 267	1843	429 911 081	11 532	9653	37 279
U_2	4679	211 082	1643	346 807 726	9303	1189	37 279
U_3	5789	170 608	4609	786 332 272	21 093	6325	37 279
$U_{1\text{-}1}$	8974	47 906	2637	126 328 122	9975	4722	12 664
$U_{1\text{-}2}$	7563	56 843	7472	424 730 896	33 538	5664	12 664
$U_{2\text{-}1}$	6637	52 253	4565	238 534 945	10 281	5464	23 201
$U_{2\text{-}2}$	6323	54 848	3822	209 629 056	9035	8021	23 201
$U_{2\text{-}3}$	6237	55 604	5578	310 159 112	13 368	8144	23 201
$U_{3\text{-}1}$	4872	161 398	1216	196 259 968	9273	6196	21 164
$U_{3\text{-}2}$	4357	180 475	2697	486 741 075	22 998	11 403	21 164

■ *Таблица 4*

Алгоритм	Время взлома, MIPS-лет	Размер ключей	Вычисление ключей	Применимость	Недостатки
J. Yeh	Задача разложения больших чисел/ $3*10^7$	Секретных — $N = 1024$ Открытых — $L = 1024$	Возведение в степень, умножение: $O(r*l*p*\ln N) = O(69{,}3*l*p)$	В любых иерархических структурах	Невозможно расширение системы, трудоемкая операция возведения в степень
Хэш-функций	Стойкость хэш-функции/ $3{,}7*10^3$	Секретных — длина блока хэш-функции, $N = 512$	Вычисление хэш столько раз, сколько уровней: $O(N*p*l) = O(512*p*l)$	В древовидных иерархических структурах	Вычисление хэш-функций столько раз, сколько уровней
Hwang	Алгоритм шифрования Triple DES/ 10^{18}	Секретных — $N = 128$ Открытых — неограничен, $L^r = 128^r$	Деление и шифрование столько раз, сколько уровней, $n = 128^r/16$: $O(p*l*(n*\ln n*\ln\ln n)) = = O(p*l*2{,}58E^{22})$	В любых иерархических структурах	Вычисление симметричного алгоритма столько раз, сколько уровней. Размер секретных ключей уменьшается в зависимости от уровня, открытых — растет
Комбинированный	Задача разложения больших чисел/ $3*10^7$	Секретных — $N = 1024$ Открытых — $L = 384$	Деление и умножение столько раз, сколько уровней: $O(p*2*l*(n*\ln n*\ln\ln n)) = = O(p*l*759)$ $n = N/16$	В древовидных иерархических структурах. Ключи группы, уровня	Размер секретных ключей незначительно уменьшается в зависимости от уровня

$$R_{1\text{-}3} = K_1 - S_{1\text{-}3}*T_{1\text{-}3} = = 429\,911\,081 - 8237*52\,192 = 5577;$$

г) вычисляет:

$$K_{1\text{-}3} = S_{1\text{-}3}*R_{1\text{-}3} = 52\,192*5577 = 291\,074\,784.$$

Шифрование и дешифрирование

Шифрование и дешифрирование происходит по ключу субъекта, которому принадлежит объект (некая информация), при помощи блочного симметричного алгоритма (например, ГОСТ, DES, AES) [1].

Например, для пользователя $U_{1\text{-}2}$ уровня 3 (см. рис. 4) администратор получает ключ пользователя следующим образом:

$$K_{1\text{-}2} = (((K_0 \text{ div } T_1)*R_1) \text{ div } T_{1\text{-}2})*R_{1\text{-}2}.$$

1) K_0 div T_1 = 987 654 321 div 4234 = > S_1 = 233 267 R_1 = 1843;

2) S_1*R_1 = 233 267*1843 = 429 911 081;

3) S_1*R_1 div $T_{1\text{-}2}$ = 429 911 081 div 7563 = > $S_{1\text{-}2}$ = 56 843 $R_{1\text{-}2}$ = 7472;

4) $S_{1\text{-}2}*R_{1\text{-}2}$ = 56 843*7472 = 424 730 896.

Таким образом, $K_{1\text{-}2}$ = 424 730 896.

Сравнительный анализ вышеприведенных алгоритмов дан в табл. 4, где r — максимальное число непосредственных потомков одного пользователя, предположим, что их 10; l — количество уровней для вычисления ключей; p — простейшая операция с 32-разрядными числами, а для реализации симметричного алгоритма шифрования и хэш-функции требуется $N*p$ операций, где N — длина ключа [14].

Заключение

Эффективный метод распределения ключей является актуальной задачей для современных компьютерных систем. Одной из областей применения данного метода могут являться системы с иерархическим доступом к информации. Так как основные требования, предъявляемые к данным системам, это возможность быстро получить ключ на любом уровне системы и безопасность системы, то предлагаемый комбинированный алгоритм удовлетворяет вышеуказанным требованиям.

Литература

1. **Гостехкомиссия** России. Руководящий документ. Средства вычислительной техники. Межсетевые экраны. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации. 1997. http://www.fstec.ru/_docs/doc_3_3_006.htm (дата обращения: 10.05.2012).
2. **Молчанов Г. А.** Защита информации в системах с разграничением полномочий // Компьютерные системы. 1999. № 3. С. 25–27.

3. **Кнут Д.** Искусство программирования для ЭВМ. — М.: Мир, 1976. Т. 1. — 729 с.
4. **Kayem Anne V. D. M., Akl S. G., Martin P.** Adaptive Cryptographic Access Control. — Springer, 2010. — 152 p.
5. **Akl S.G., Taylor P. D.** Cryptographic solution to a problem of access control in a hierarchy // ACM Transactions on Computer Systems. 1983. N 1(3). P. 239–248.
6. **Chen H.-C., Wang S.-J., Wen J.-H.** A Cryptographic Key Assignment Scheme with Adaptable Timetoken Constraint in a Hierarchy // Intern. J. of Multimedia Ubiquitous Engineering. Oct. 2008. Vol. 3. N 4. P. 521–526.
7. **Bertino E., Shang N., Wagstaff Samuel S. Jr.** An Efficient Time-Bound Hierarchical Key Management Scheme for Secure Broadcasting // IEEE Trans. on Dependable and Secure Computing. 2008. Vol. 5. N 2. P. 65–70.
8. **Shyam R.** An Efficient Time-Bound Hierarchical Key Assignment Scheme With a New Merge Function // A Performance Study. Dec. 2009. P. 26–35. http://scholarworks.boisestate.edu/cs_gradproj/1/ (дата обращения: 24.08.2012).
9. **Sandhu R. S.** Cryptographic implementation of a tree hierarchy for access control // Information Processing Letters. 1988. N 27(2). P. 95–98.
10. **Лёвин В. Ю.** О повышении криптостойкости однонаправленных хэш-функций // Фундаментальная и прикладная математика. 2009. Т. 15. № 5. С. 171–179.
11. **Hwang M.-S.** A New Dynamic Cryptographic Key Generation Scheme for a Hierarchy // Nordic J. of Computing. Aug. 1999. Vol. 6. N 4. P. 363–371.
12. **Лернер В. Д., Беззатеев С. В.** Основные принципы распределения ключей для доступа к информации в «облачных» хранилищах данных // Информационная безопасность регионов России (ИБРР-2011): VII Санкт-Петербургская межрегион. конф., Санкт-Петербург, 26–28 октября 2011 г.: материалы конф. / СПОИСУ. СПб., 2011. С. 120.
13. **Лернер В. Д.** Использование операции арифметического деления для распределения ключей: сб. докл. второй науч. конф. ГУАП / ГУАП. СПб., 1999. С. 40.
14. **Винокуров А., Применко Э.** Сравнение российского стандарта шифрования, алгоритма ГОСТ 28147-89 и алгоритма Rijndael // Системы безопасности связи и телекоммуникаций. 2001. № 39(3)/01. С. 71–72.