Обучающая методичка по алгоритму Эль Гамаля

Схема Эль-Гамаля — это криптосистема с открытым ключом, основанная на трудности вычисления дискретных логарифмов в конечном поле. Криптосистема включает в себя алгоритм шифрования и алгоритм цифровой подписи. Схема была предложена Тахером Эль-Гамалем в 1985 году. Он усовершенствовал систему Диффи-Хеллмана и получил два алгоритма, которые использовались для шифрования и для обеспечения аутентификации. В отличие от RSA алгоритм Эль-Гамаля не был запатентован и, поэтому, стал более дешевой альтернативой.

Что такое криптосистема с открытым и закрытым ключом (или как её ещё называют – ассиметричное шифрование)?

Это такая система, где чтобы зашифровать сообщение используется один ключ, а чтобы расшифровать – второй, в отличие от симметричного шифрования, где используется один и тот же ключ для шифрования и расшифровки. Здесь эти ключи называют так: первый – это открытый (или публичный), а второй – закрытый (или секретный). Открытый ключ рассказывается кому угодно, с помощью него сообщение шифруется и передаётся обладателю этого ключа. Далее он (обладатель) расшифровывает сообщение с помощью секретного ключа.

Теперь сам алгоритм:

Первым делом нужно сгенерировать числа P и A ($1 < A < P - 1$) такие, что P простое, а A является генератором мультипликативной группы кольца вычетов по модулю P (*или как ещё это называют: A – это первообразный корень по модулю P*).

Что такое генератор мультипликативной группы кольца вычетов по модулю P? В данном случае это число A такое, что все числа из интервала [1, 2, 3, …, P-1] могут быть представлены как различные степень A mod(P). Рассмотрим на примере:

Возьмём группу Z7: {0, 1, 2, 3, 4, 5, 6}. Недолго думая, выберем число 3. Проверим:

$3^1 = 3 \bmod(7)$ | $3^4 = 4 \bmod(7)$

$3^2 = 2 \bmod(7)$ | $3^5 = 5 \bmod(7)$

$3^3 = 6 \bmod(7)$ | $3^6 = 1 \bmod(7)$

Видно, что все числа {1, 2, 3, 4, 5, 6} представляются в виде (3^i)mod(7), где i от 1 до 6 {1= 3^6, 2=3^2, 3=3^1, 4=3^4, 5= 3^5, 6=3^3}. Значит 3 – это генератор мультипликативной группы кольца вычетов по модулю 7. Но это только пример, здесь 3 и 7 – очень маленькие числа. На самом деле на практике берутся числа, у которых хотя бы 250-300 цифр в записи.

Чтобы было просто генерировать такие числа можно использовать следующий подход: простое число P берётся такое, что P = 2*q+1, где число q тоже простое. Тогда в качестве A можно взять число, для которого выполняется: $A^q \bmod(P) \neq 1$ и $1 < A < P - 1$. Такой выбор числа P также и усиливает криптостойкость.

Вторым шагом каждый пользователь выбирает себе секретный ключ X и публичный ключ Y следующий образом:

X такое, что: $1 < X < P-1$

Y такое, что: $Y = A^X \bmod(P)$

Будем обозначать X_1 и Y_1 секретный и публичный ключи для пользователя 1 и X_2 и Y_2 для пользователя 2 соответственно, и так далее.

По сути, теперь уже можно начинать шифровать и дешифровать сообщения. А также их подписывать.

Введём ещё одно обозначение. Будем обозначать числом m само сообщение. Это число может быть получено разными способами, главное, чтобы оно не было бы больше, чем P. Хотя бы можно поступать так: какое-то

сообщение представляется в двоичном виде (*каждую букву представить как байт, а потом просто записать все эти байты один за другим*), этот же двоичный вид может представлять число m. Ну а если это число m получается больше, чем P, то сообщение можно “разбить” на более мелкие.

Добавим ещё одно обозначение: операцию деления по модулю mod. Другими словами, она находит просто остаток от деления. Например, 24mod(10) = 4 или 13mod(11) = 2, или 5mod(2) = 1 и т. д.

Шифрование, передача и расшифровка:

Пользователь 1 хочет отправить сообщение m в зашифрованном виде.

Шаг 1: пользователь 2 передаёт пользователю 1 числа P, A и Y_2.

Шаг 2: пользователь 1 находит числа k, r и e:

k такое, что наибольший общий делитель k и P-1 равен 1.

$r = A^k \bmod(P)$

$e = (m*Y_2^k) \bmod(P)$

Теперь зашифрованное сообщение представляется в таком виде (r, e). То есть зашифрованное сообщение (r, e) передаётся пользователю 2.

Шаг 3: пользователь 2 расшифровывает:

$m = (e*r^{(P-1-X2)}) \bmod(P)$

Почему это вообще работает?!

Начнём с теоремы Эйлера. Она гласит:

Если a и P взаимно просты (то есть их наибольший общий делитель равен 1), то $a^{\varphi(P)} \bmod(P) = 1$, где φ(P) – это функция Эйлера. Теперь про функцию Эйлера – это функция, которая равна количеству натуральных чисел, меньших P и взаимно с простых с P. Например, φ(24) = 8, так как всего 8 чисел, меньших 24, взаимно просты с 24 (*т. е. имеют наибольший общий делитель 1*): 1, 5, 7,

11, 13, 17, 19, 23. Нетрудно догадаться, что если P – простое, то все числа, меньшие P, будут взаимно просты с P. Т.е. получается, что если a не делится на простое число P, то $a^{(P-1)} \bmod(P) = 1$ (это называется малой теоремой Ферма).

Так вот, говорилось что:

$m = (e*r^{(P-1-X2)}) \bmod(P)$, где $e = (m*Y_2^k) \bmod(P)$, $r = A^k \bmod(P)$ и $Y_2 = A^{X2} \bmod(P)$. Подставим e, r и Y_2 в $(e*r^{(P-1-X2)}) \bmod(P)$:

$m = ((m*Y_2^k) * A^{k*(P-1-X2)}) \bmod(P) = ((m*A^{k*X2}) * A^{k*(P-1-X2)}) \bmod(P)$

Сразу возникает вопрос, а так вообще можно делать? Ну подставлять, например, это вот (a*3)mod(P) = c сюда (31*c*b)mod(P) так, что уберётся первый mod: (31*(a*3)mod(P) * b)mod(P). Так вот, так делать можно (проверьте сами на каком-нибудь примере), и получится:

(31*c*b)mod(P) = (31* (a*3)mod(P) * b)mod(P) = (93 * a * b)mod(P)

Продолжим:

$((m*A^{k*X2}) * A^{k*(P-1-X2)}) \bmod(P) = ((m*A^{k*X2+ k*(P-1-X2)}) \bmod(P) =$

$((m*A^{k*(P-1)}) \bmod(P) =$ [обозначим $t = A^k$] = $((m*t^{(P-1)}) \bmod(P) =$ [по теореме выше, так как P изначально простое, $t^{(P-1)} \bmod(P) = 1$] = $((m*1) \bmod(P) = m$. Действительно, m = m.

Ладно, звучит всё хорошо, но что, если мы будем полностью слушать канал, в котором передаются все эти сообщения? Тогда мы узнаем: P, A, e, r и Y2. Но как ни крути раскрыть m не получится, у нас не хватает X_2 и k. Подбирать их будем до конца вселенной, если, конечно, речь идёт о больших числах.

Теперь подпись сообщений. Или как её в данном случае лучше назвать – электронная подпись. Она позволяет подписать сообщение так, что если его чуть-чуть изменить, то сразу станет понятно, что это (чуть-чуть изменённое сообщение) вы не подписывали.

Но сначала нужно понять, что такое хеш-функция. Это такая функция f(x) = y, что у вычислить очень легко, а вот x найти, зная y, очень сложно. Скажем вместо x подставляется набор байт (сколь угодно длинный), который на выходе даёт (ровно N байт) = y. Причём если в x поменять где-нибудь хотя бы 1 бит, то y изменится настолько, что его даже x не узнает. В общем смысл понятен. Например, такой функцией может быть SHA-512 или SHA-256, MD5 и т.д. Ну или вообще такое: f(a, b) = y, где, для наглядности, a и b простые, а y = a*b. Здесь y вычислить легко, а вот восстановить a и b сложно.

Подпись сообщений.

Пользователь хочет подписать сообщение M (*это вполне может быть документ или просто строка символов*).

Шаг 1: Пользователь пользуется хэш-функцией f(M) = m. На выходе получается в данном случае число.

Шаг 2: Пользователь вычисляет числа k, r и e:

k такое, что $1 < k < P-1$ и взаимно простое с P-1

$r = A^k \bmod(P)$

$e = ((m – X*r)*k^{-1}) \bmod(P-1)$, тут k^{-1} – это мультипликативное обратное. Что это такое и как его найти скажу ниже. Также обращаю внимание на “mod(P-**1**)”.

Теперь кому-то (кому надо) передаётся само сообщение M, P, A и (r, e), где (r, e) и есть подпись сообщения.

Шаг3: Проверка подписи. Снова вычисляется m = f(M). Далее если выполняется условие: $(Y^r * r^e) \bmod(P) = A^m \bmod(P)$ - то подпись верна.

Замечание: здесь очень важно, чтобы было действительно сложно вычислить x, зная y=f(x). Ведь можно меняя биты в x рано или поздно найти такое x`, что f(x`) = f(x), так как y=f(x) фиксированной длины, а x – нет.

Замечание: важно, чтобы k было одноразовым для каждого M, так как можно (если всё-таки подобрать k) узнать $X = ((m - k*e)*r^{-1}) \bmod(P-1)$, а зная X, можно подделать подпись, да и вообще расшифровывать сообщения.

Почему это всё верно? Было сказано, что $e = ((m - X*r)*k^{-1}) \bmod(P-1)$. Выразим отсюда $m = (e*k + X*r) \bmod(P-1)$ (да, так тоже можно делать). Замечу (для примера), что если $(a)\bmod(P) = b$, то $a = P*c + b$, где c – какое-то целое число. Действительно, выражение $(a = P*c + b)$ показывает, что b – это остаток от деления a на P (изначально говорилось, что mod – это остаток). Частное c сейчас нас не сильно волнует. Тогда получается, что $m = (P-1)*c + (e*k + X*r)$.

Теперь $A^m \bmod(P) = A^{(P-1)*c + (e*k + X*r)} \bmod(P) = (A^{c*(P-1)} * A^{(e*k + X*r)}) \bmod(P) =$ [сделаем замену $t = A^c$] $= (t^{(P-1)} * A^{(e*k + X*r)}) \bmod(P) =$ [по малой теореме Ферма или по теореме Эйлера: $t^{(P-1)} \bmod(P) = 1$, т.к. P простое] $= (A^{(e*k + X*r)}) \bmod(P) = (A^{e*k} * A^{X*r}) \bmod(P) =$ [вспомним, что $A^k \bmod(P) = r$ и $A^X \bmod(P) = Y$] =

$(Y^r * r^e) \bmod(P)$. В общем, что и требовалось доказать.

Замечу, что вместо M можно использовать какой-нибудь ключ, тем самым подписать его. Но об этом позже.

Теперь вернёмся к вопросу: “Что такое мультипликативное обратное и как его искать?”. Если совсем прямо, то $(k^{-1})\bmod(P) = (1/k)\bmod(P)$. Лучше это рассмотреть на примере: $(1/9)\bmod(7) = (x)\bmod(7)$. Нужно найти x. Умножим правую и левую часть на 9: $(1)\bmod(7) = 1 = (9*x)\bmod(7)$. Другими словами, $1= 7*c + 9*x$, где c – какое-то целое число.

Вот это вот: $1= 7*c + 9*x$ – называется диофантово уравнение. Решается оно так:

$9 = 7*1 + 2$ | отсюда выразим 2 + подставим в нижнее: $1 = 7- (9-7)*3 = 7*2-3*9$

$7 = 2*3 + 1$ | отсюда выразим $1= 7-2*3$

$2 = 1*2 + 0$

Имеем: 1= 7*c + 9*x и 1 = 7*2-3*9. Отсюда x = (-3) mod(7) = (-3+7) mod(7) = 4. Замечу, что 7 mod(7) = 0, а нуль можно сколько угодно раз прибавлять. Действительно:

(9*(1/9)) mod(7) = 1 = (9*x)mod(7) = (9*4) mod(7) = (36) mod(7) = 1.

Короче (1/9) mod(7) = 4. Вот так это находится.

Всё звучит хорошо. Но как мы узнаем, что к нам пришёл публичный ключ именно того пользователя, от которого мы хотим (*его вообще-то могут поменять, пока он к вам "идёт", и тогда вы будете общаться с кем-то вообще другим*)? И вообще, как узнать публичный ключ нужного нам пользователя?

Вот постановка задачи: Майк хочет получить публичный ключ Антона так, чтобы гарантировать, что это именно его публичный ключ. Скажем, что Майк и Антон никак не могут встретиться лично. Здесь в игру вступает Пётр (или центр сертификации).

Цифровые сертификаты:

Цифровым сертификатом будем называть подписанную кем-то документ, в котором содержится публичный ключ, его описание и данные владельца этого публичного ключа.

Шаг 0: Пётр всем как-нибудь рассказывает свой публичный ключ. По радио, по телевизору, на заборе, в рекламе или ещё как-то.

Шаг 1: Пётр каким-то образом удостоверяет личность Антона (*например, Пётр зарабатывает на жизнь тем, что он ездит к людям и лично с ними встречается, тем самым он удостоверяется в личности этих пользователей, или пользователи сами приезжает к Петру*) и получает публичный ключ Антона.

Шаг 2: Пётр подписывает публичный ключ Антона + добавляет к нему какой-нибудь описание. Например, имя и почту Антона. Или ещё пример:

добавляет описание, что Пётр полностью доверяет Антону, и Антон может подписывать другие ключи также, как Пётр.

Шаг 3: Пётр отсылает Антону его теперь уже подписанный ключ.

Шаг 4: теперь Антон может отослать Майку свой подписанный публичный ключ или Майк сам может запросить подписанный ключ у Петра (*всё же ведь Пётр подписывает публичную информацию, поэтому он предоставит Майку ключ*).

Шаг 5: Майк смотрит, что этот подписанный ключ был подписан именно Петром (*для этого Майк использует публичный ключ Петра, который был всем рассказан в шаге 0*).

Замечание: публичный ключ Петра может быть подписан другим пользователем, доверие к которому больше и так далее.

Теперь давайте проанализируем всю эту ситуацию. Если в подписанном ключе изменить хоть 1 бит, то подпись сразу станет не верна. Так что можно вполне быть уверенным, что никто не изменил подписанный ключ, пока он шёл от Петра к Антону или Майку или от Антона к Майку.

Вообще, чтобы зашифровать сообщение с помощью алгоритма Эль-Гамаля, нужны большие вычислительные мощности. А если сообщений много? Намного разумнее будет использовать алгоритм Эль-Гамаля, чтобы обменяться каким-нибудь ключом для симметричного шифрования. Например, для AES-256 или тому подобных.

Использованные источники:

1) Басалова Г.В. Основы криптографии. 2016. - С. 156-233.

2) Б.Я.Рябко, А.Н.Фионов КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ. - Москва: Горячаялиния Телеком, 2005. - С. 31-34, 41-50.

3) TAHER ELGAMAL A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms // IEEE TRANSACTIONS ON INFORMATION THEORY. - JULY 1985. - №4.

4) Rabin M. O. Probabilistic algorithm for testing primality // JOURNAL OF NUMBER THEORY . - 1980. - №12. - С. 128-138.

5) Lee C. H., Lee P. J A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroup