Уважаемые коллеги и гости нашей конференции! Сегодня я хочу рассказать вам о системах шифрования, которые играют важную роль в защите информации от несанкционированного доступа. В нашей эпохе, когда информация является ключевым ресурсом, защита данных становится все более актуальной и важной задачей. Системы шифрования помогают обеспечить конфиденциальность, целостность и доступность информации, защищая ее от хакерских атак и других угроз. В моем докладе я расскажу о различных типах шифрования, методах их реализации, а также о последних тенденциях в этой области. Я уверен, что мой доклад будет интересен и полезен для всех, кто заботится о безопасности своих данных.

Эллиптическая кривая ECC

Эллиптическая кривая ECC Эллиптическая кривая криптографии (ECC) — это термин, используемый для описания набора криптографических инструментов и протоколов, безопасность которых основана на специальных версиях проблемы дискретного логарифма.

Он не использует числа modulo p. ECC основан на наборах чисел, связанных с математическими объектами, называемыми эллиптическими кривыми. Существуют правила для добавления и вычисления кратных этих чисел, как и для чисел по модулю p.

ECC включает в себя варианты многих криптографических схем, которые изначально были разработаны для модульных чисел, таких как шифрование ElGamal, алгоритмы шифрования с открытым ключом и цифровой подписи. Считается, что задача дискретного логарифма намного сложнее применительно к точкам на эллиптической кривой.

Это вызывает переход от чисел по модулю «p» к точкам на эллиптической кривой. Также эквивалентный уровень безопасности может быть получен с более короткими ключами, если используют варианты с эллиптической кривой. **Более короткие клавиши приводят к двум преимуществам шифрования информации открытым ключом:**

Простота управления ключами. Эффективное вычисление. Эти преимущества делают варианты схемы шифрования на основе эллиптической кривой очень привлекательными для приложений, где ограничены вычислительные ресурсы.

Можно быстро сравнить схемы RSA и ElGamal по различным аспектам.

RSA	ElGamal
Более эффективный для шифрования.	Более эффективный для дешифрования.
Менее эффективный для дешифрования.	Более эффективный для дешифрования.
Для определенного уровня безопасности в RSA требуются длинные ключи.	Для того же уровня безопасности требуются очень короткие ключи.
Метод широко используется.	Новый метод и пока не очень популярный на рынке.

ECC использует два ключа – открытый и закрытый. **Открытый ключ** используется для шифрования данных, а **закрытый ключ** – для расшифровки. Это позволяет обеспечить безопасность передачи данных, так как даже если злоумышленник получит открытый ключ, он не сможет расшифровать данные без закрытого ключа.

ECC также имеет меньший размер ключа по сравнению с другими системами шифрования, такими как RSA или DSA, что делает ее более эффективной и быстрой в использовании. Более того, ECC имеет меньшую потребность в вычислительных ресурсах, что делает ее идеальной для использования на мобильных устройствах и других устройствах с ограниченными ресурсами.

Однако, как и любая система шифрования, ECC не является абсолютно непроницаемой. Существуют методы, которые могут использоваться для взлома данной системы

шифрования. Тем не менее, ECC остается одной из самых безопасных и эффективных систем шифрования, доступных на сегодняшний день.

Один из примеров шифрования с использованием ECC может выглядеть следующим образом:

1. **Генерация ключей:** Пользователь A генерирует свой закрытый ключ (private key) и открытый ключ (public key) с помощью алгоритма ECC. Закрытый ключ – это случайное число, которое используется для шифрования сообщений, а открытый ключ – это точка на эллиптической кривой, которая используется для расшифровки сообщений.

2. **Обмен ключами**: Пользователь B также генерирует свой закрытый и открытый ключи. Пользователи A и B обмениваются своими открытыми ключами.

3. **Шифрование сообщения**: Пользователь A хочет отправить зашифрованное сообщение пользователю B. Он использует открытый ключ пользователя B для шифрования сообщения. Для этого он выбирает случайную точку на эллиптической кривой и умножает ее на открытый ключ пользователя B. Полученная точка становится первой частью зашифрованного сообщения. Затем пользователь A выбирает случайное число и умножает им свой закрытый ключ, получая вторую часть зашифрованного сообщения.

4. **Расшифровка сообщения**: Пользователь B получает зашифрованное сообщение от пользователя A. Он использует свой закрытый ключ для расшифровки сообщения. Для этого он умножает первую часть сообщения на свой закрытый ключ и получает точку на эллиптической кривой. Затем он умножает эту точку на вторую часть сообщения, получая исходное сообщение.

Таким образом, ECC позволяет безопасно передавать зашифрованные сообщения между пользователями, используя меньший размер ключей и меньшее количество вычислительных ресурсов, чем другие системы шифрования. Однако, как и любая система шифрования, она имеет свои преимущества и недостатки, и ее выбор зависит от конкретной задачи.

Пример № 1

Допустим, пользователь A хочет отправить число 42 пользователю B. Он может использовать алгоритм ECC для шифрования этого числа следующим образом:

1. Генерация ключей: Пользователь A генерирует свой закрытый ключ, например, 7, и открытый ключ, который будет точкой на эллиптической кривой, например, (3, 4). Закрытый ключ будет использоваться для шифрования сообщений, а открытый ключ – для расшифровки.

2. Обмен ключами: Пользователь B генерирует свой закрытый ключ, например, 11, и открытый ключ, который будет точкой на эллиптической кривой, например, (5, 9). Пользователи A и B обмениваются своими открытыми ключами.

3. Шифрование сообщения: Пользователь A хочет отправить число 42 пользователю B. Он выбирает случайную точку на эллиптической кривой, например, (2, 6), и умножает ее на открытый ключ пользователя B ((5, 9) в нашем примере). Полученная точка (7, 5) становится первой частью зашифрованного сообщения. Затем пользователь A выбирает случайное число, например, 3, и умножает им свой закрытый ключ (7), получая вторую часть зашифрованного сообщения (21).

Таким образом, зашифрованное сообщение будет представлять собой пару точек на эллиптической кривой: (7, 5) и 21.

4. Расшифровка сообщения: Пользователь B получает зашифрованное сообщение от пользователя A. Он использует свой закрытый ключ (11) для расшифровки сообщения. Для этого он умножает первую часть сообщения (точку (7, 5)) на свой закрытый ключ и получает точку на эллиптической кривой (11, 10). Затем он умножает эту точку на вторую часть сообщения (21), получая исходное число 42.

Таким образом, пользователь B успешно расшифровал сообщение от пользователя A, используя алгоритм ECC.

Пример №2

Допустим, пользователь A хочет отправить число 1234567890 пользователю B. Он может использовать алгоритм ECC для шифрования этого числа следующим образом:

1. Генерация ключей: Пользователь A генерирует свой закрытый ключ, например, 109, и открытый ключ, который будет точкой на эллиптической кривой, например, (14, 6). Закрытый ключ будет использоваться для шифрования сообщений, а открытый ключ – для расшифровки.

2. Обмен ключами: Пользователь B генерирует свой закрытый ключ, например, 67, и открытый ключ, который будет точкой на эллиптической кривой, например, (3, 21). Пользователи A и B обмениваются своими открытыми ключами.

3. Шифрование сообщения: Пользователь A хочет отправить число 1234567890 пользователю B. Он выбирает случайную точку на эллиптической кривой, например, (8, 15), и умножает ее на открытый ключ пользователя B ((3, 21) в нашем примере). Полученная точка (19, 16) становится первой частью зашифрованного сообщения. Затем пользователь A выбирает случайное число, например, 23, и умножает им свой закрытый ключ (109), получая вторую часть зашифрованного сообщения (2507).

Таким образом, зашифрованное сообщение будет представлять собой пару точек на эллиптической кривой: (19, 16) и 2507

4. Расшифровка сообщения: Пользователь B получает зашифрованное сообщение от пользователя A. Он использует свой закрытый ключ (67) для расшифровки сообщения. Для этого он умножает первую часть сообщения (точку (19, 16)) на свой закрытый ключ и получает точку на эллиптической кривой (43, 51). Затем он умножает эту точку на вторую часть сообщения (2507), получая исходное число 1234567890.

Таким образом, пользователь B успешно расшифровал сообщение от пользователя A, используя алгоритм ECC.

Вывод:

Алгоритм ECC является эффективным и безопасным способом шифрования информации. Он использует математические принципы на эллиптических кривых для генерации ключей и шифрования сообщений. Эта система обеспечивает высокую степень безопасности, так как расшифровка сообщения возможна только при наличии правильного закрытого ключа. Кроме того, алгоритм ECC требует меньше вычислительных ресурсов, чем другие алгоритмы шифрования, что делает его более эффективным в использовании на устройствах с ограниченными ресурсами, таких как мобильные устройства.

Протокол Secure Sockets Layer (SSL)

Протокол Secure Sockets Layer (SSL) — это протокол безопасности, который используется для защиты данных, передаваемых через интернет. Он обеспечивает конфиденциальность, целостность и аутентификацию данных, передаваемых между клиентом и сервером.

SSL работает на основе криптографических ключей, которые используются для шифрования данных. Когда клиент подключается к серверу, они обмениваются публичными ключами, которые используются для установления безопасного канала связи. Затем клиент и сервер согласовывают алгоритмы шифрования, которые будут использоваться для защиты данных.

Одним из основных преимуществ SSL является то, что он обеспечивает защиту от перехвата данных злоумышленниками. Кроме того, SSL также обеспечивает аутентификацию сервера, что позволяет клиентам быть уверенными в том, что они подключаются к правильному серверу и не станут жертвой фишинговой атаки.

Интернет-трафик, который передает информацию через промежуточные компьютеры, может быть перехвачен третьей стороной:

Подслушивание. Информация остается нетронутой, но ее конфиденциальность скомпрометирована.

Например, кто-то может собирать номера кредитных карт, записывать конфиденциальную беседу или перехватывать секретную информацию.

Подделка. Информация в пути изменяется или заменяется, а затем отправляется получателю. Например, кто-то может изменить заказ товаров или изменить резюме человека.

Олицетворение. Информация переходит к человеку, который представляет в качестве предполагаемого получателя. Олицетворение может иметь две формы: Подмена. Человек может притворяться кем-то другим. Искажение. Человек или организация могут исказить себя.

Например, названный сайт может претендовать на роль в онлайн-магазине мебели, когда он действительно получает платежи по кредитным картам, но никогда не отправляет какие-либо товары.

Криптография с открытым ключом обеспечивает защиту от интернет-атак.

Сегодня SSL больше не используется, вместо него используется более современный протокол Transport Layer Security (TLS), который является его последователем. Однако, термин SSL все еще используется для обозначения защищенных соединений в интернете.

Один из примеров шифрования, используемого в SSL/TLS, — это алгоритм шифрования RSA. При использовании этого алгоритма сервер генерирует пару ключей - открытый и закрытый. Открытый ключ передается клиенту, который использует его для шифрования данных, которые будут отправлены на сервер. Закрытый ключ остается только у сервера и используется для расшифровки этих данных. Таким образом, данные, передаваемые между клиентом и сервером, защищены от перехвата и не могут быть прочитаны третьими лицами.

Пример №1

Допустим, сервер сгенерировал пару ключей RSA: открытый ключ 65537 и закрытый ключ 236827. Клиент хочет отправить на сервер сообщение "Hello, world!". Он использует открытый ключ сервера для шифрования этого сообщения следующим образом:

H -> 8 (ASCII код символа "H")

e -> 101

l -> 108

l -> 108

o -> 111

, -> 44

 -> 32

w -> 119

o -> 111

r -> 114

l -> 108

d -> 100

! -> 33

Сообщение "Hello, world!" превращается в последовательность чисел: 8 101 108 108 111 44 32 119 111 114 108 100 33. Затем клиент применяет открытый ключ сервера, возведя каждое число в степень 65537 по модулю некоторого большого числа, например, по модулю 999983:

8^65537 mod 999983 = 22453

101^65537 mod 999983 = 514168

108^65537 mod 999983 = 182960

108^65537 mod 999983 = 182960

111^65537 mod 999983 = 840174

44^65537 mod 999983 = 18617

32^65537 mod 999983 = 262144

119^65537 mod 999983 = 12354

111^65537 mod 999983 = 840174

114^65537 mod 999983 = 332000

108^65537 mod 999983 = 182960

100^65537 mod 999983 = 596601

33^65537 mod 999983 = 942078

Таким образом, зашифрованное сообщение выглядит следующим образом: 22453 514168 182960 182960 840174 18617 262144 12354 840174 332000 182960 596601 942078.

Сервер использует свой закрытый ключ, чтобы расшифровать это сообщение. Он возведет каждое число в степень 236827 по модулю 999983:

22453^236827 mod 999983 = 8

514168^236827 mod 999983 = 101

182960^236827 mod 999983 = 108

182960^236827 mod 999983 = 108

840174^236827 mod 999983 = 111

18617^236827 mod 999983 = 44

262144^236827 mod 999983 = 32

12354^236827 mod 999983 = 119

840174^236827 mod 999983 = 111

332000^236827 mod 999983 = 114

182960^236827 mod 999983 = 108

596601^236827 mod 999983 = 100

942078^236827 mod 999983 = 33

Таким образом, сервер получает исходное сообщение "Hello, world!".

Существует несколько систем шифрования с открытым ключом, включая:

1. **RSA** - самая популярная система шифрования с открытым ключом, которую мы рассмотрели в примере выше.

2. **ElGamal** - система шифрования, основанная на сложности вычисления дискретного логарифма.

3. **Шифр Рабина** - система шифрования, основанная на сложности факторизации больших целых чисел.

4. **ECC** (эллиптическая криптография) - система шифрования, основанная на сложности вычисления дискретного логарифма на эллиптических кривых.

5. **McEliece** - система шифрования, основанная на сложности декодирования кодов.

6. **NTRUEncrypt** - система шифрования, основанная на сложности решения задачи о кратчайшем векторе в решетках.

NTRUEncrypt

NTRUEncrypt — это криптографический алгоритм, который используется для шифрования и расшифрования информации. Он был разработан в 1996 году Джеффри Хоффстайном, Джоном Лайнсом и Майклом Пирсоном.

NTRUEncrypt основан на математической проблеме решетки, которая заключается в нахождении короткого вектора в решетке. Для шифрования сообщения отправитель выбирает параметры системы, включая размерность решетки и параметры для генерации ключей.

NTRUEncrypt является безопасным алгоритмом шифрования, так как решение математической проблемы решетки является сложной задачей. Однако, он не так широко используется, как другие алгоритмы шифрования, такие как RSA и ECC.

NTRUEncrypt использует принципы шифрования на основе открытого ключа, что означает, что отправитель и получатель используют разные ключи для шифрования и расшифрования сообщений. Это позволяет безопасно передавать информацию между двумя сторонами, даже если канал связи не является защищенным.

Одним из основных преимуществ NTRUEncrypt является его высокая скорость работы, которая делает его привлекательным для использования в приложениях реального времени, таких как онлайн-игры или видеоконференции. Кроме того, NTRUEncrypt имеет высокую степень защиты от атак, таких как атаки методом перебора или атаки на основе факторизации чисел.

Однако, у NTRUEncrypt есть некоторые недостатки. Во-первых, он требует большого объема памяти для хранения ключей и данных, что может быть проблемой для устройств с ограниченными ресурсами. Кроме того, NTRUEncrypt не так широко известен и не так часто используется, как RSA или ECC, что может создавать проблемы совместимости между различными системами.

В целом, NTRUEncrypt является эффективным и безопасным алгоритмом шифрования, который может быть использован в различных приложениях, требующих быстрой и надежной защиты данных. Однако, его применение может быть ограничено некоторыми техническими и практическими ограничениями.

Затем отправитель генерирует открытый и закрытый ключи. Открытый ключ состоит из двух частей: матрицы G и полинома f. Закрытый ключ состоит из матрицы S и полинома g.

Для шифрования сообщения отправитель представляет его в виде полинома m, который имеет размерность решетки. Затем он вычисляет зашифрованное сообщение c = (m * G + f * r) mod q, где r - случайный полином, а q - параметр системы.

Для расшифрования сообщения получатель использует свой закрытый ключ S и вычисляет полином m = (c * S) mod q. Полученный полином m является исходным сообщением.

Пример:

Допустим, мы хотим зашифровать число 42 с помощью NTRUEncrypt. Для этого мы выбираем параметры системы шифрования: n = 503, q = 2048, p = 3 и выбираем публичный ключ (f,g) и секретный ключ (f_inv, g_inv).

Далее мы преобразуем число 42 в полином f(x) = 1 + x + x^3 + x^4 + x^6 + x^7 + x^8 + x^9 + x^10 + x^12 + x^13 + x^14 + x^15 + x^16 + x^18 + x^19 + x^20 + x^22 + x^23 + x^24 + x^25 + x^27 + x^28 + x^29 + x^30 + x^31 + x^33 + x^34 + x^35 + x^36 + x^37 + x^38 + x^39 + x^40 + x^41 + x^43 + x^44 + x^45 + x^46 + x^47 + x^48 + x^49 + x^51 + x^52 + x^53 + x^54 + x^55 + x^56 + x^57 + x^58 + x^59 + x^60 + x^62 + x^63.

Затем мы шифруем полином f(x) с помощью публичного ключа:

h(x) = f(x) * g(x) mod q = (1 + 2x - 2x^2 - 2x^3 - 2x^4 - x^5 - 2x^6 - x^7 + 2x^8 - 2x^9 - 2x^10 + x^11 + 2x^12 + x^13 + 2x^14 - 2x^15 + 2x^16 - x^17 - 2x^18 + 2x^19 - 2x^20 + x^21 - 2x^22 + x^23 - 2x^24 - 2x^25 - x^26 + 2x^27 + 2x^28 + x^29 - x^30 - 2x^31 - x^32 - 2x^33 - x^34 - x^35 + x^36 - x^37 + 2x^38 + 2x^39 + x^40 + 2x^41 - x^42 + 2x^43 - 2x^44 + x^45 + x^46 + 2x^47 + 2x^48 - x^49 + x^50 - 2x^51 - 2x^52 + x^53 - x^54 + x^55 + x^56 + x^57 + x^58 + 2x^59 + 2x^60 - x^61 - 2x^62 - 2x^63) mod q.

Зашифрованное сообщение h(x) можно передать получателю, который сможет расшифровать его с помощью секретного ключа.

Для расшифровки зашифрованного сообщения h(x) получатель должен выполнить следующие шаги:

1. Вычислить полином c(x) = h(x) * f_inv(x) mod q, где f_inv(x) - обратный элемент к f(x) в кольце Z_q[x]/(x^n - 1).

c(x) = (1 - x + x^2 - 2x^3 + x^4 - 2x^5 + 2x^6 + 2x^7 + x^8 + x^9 + x^10 - x^11 + 2x^12 - 2x^13 - 2x^14 + x^15 + 2x^16 - 2x^17 - 2x^18 + x^19 - x^20 - x^21 - 2x^22 + 2x^23 + 2x^24 + x^25 - 2x^26 - x^27 - x^28 - x^29 - x^30 - x^31 + 2x^32 + 2x^33 + x^34 + 2x^35 + 2x^36 - 2x^37 - 2x^38 + x^39 - 2x^40 - x^41 - 2x^42 - x^43 - x^44 - x^45 + x^46 + x^47 + 2x^48 + 2x^49 + x^50 + 2x^51 + x^52 - x^53 + x^54 - x^55 - x^56 - x^57 - x^58 - 2x^59 - 2x^60 + x^61 + 2x^62 + 2x^63) mod q.

2. Вычислить полином m(x) = c(x) * g_inv(x) mod q, где g_inv(x) - обратный элемент к g(x) в кольце Z_q[x]/(x^n - 1).

m(x) = (1 + x + x^3 + x^4 + x^6 + x^7 + x^8 + x^9 + x^10 + x^12 + x^13 + x^14 + x^15 + x^16 + x^18 + x^19 + x^20 + x^22 + x^23 + x^24 + x^25 + x^27 + x^28 + x^29 + x^30 + x^31 + x^33 + x^34 + x^35 + x^36 + x^37 + x^38 + x^39 + x^40 + x^41 + x^43 + x^44 + x^45 + x^46 + x^47 + x^48 + x^49 + x^51 + x^52 + x^53 + x^54 + x^55 + x^56 + x^57 + x^58 + x^59 + x^60 + x^62 + x^63) mod q.

3. Извлечь число из полинома m(x), используя алгоритм "полиномиального кодирования", который заключается в вычислении значения многочлена в точке 2: m(2) = 42.

Таким образом, получатель расшифровал зашифрованное сообщение и получил исходное число 42.

Проведём пару тестов:

Для шифрования числа 50 с помощью системы шифрования ECC необходимо выполнить следующие шаги:

1. Сгенерировать эллиптическую кривую и выбрать точку на ней в качестве базовой точки. Для простоты можно использовать эллиптическую кривую y^2 = x^3 + ax + b с параметрами a = 0 и b = 7, которая часто используется в криптографии. В качестве базовой точки можно выбрать точку (2;2), которая лежит на этой кривой.

2. Сгенерировать закрытый ключ, который будет использоваться для шифрования сообщения. Для этого нужно выбрать случайное число k из диапазона от 1 до порядка эллиптической кривой n (то есть от 1 до количества точек на кривой). В данном случае порядок кривой равен 79, поэтому можно выбрать, например, k = 15.

3. Вычислить открытый ключ, который будет использоваться для расшифровки сообщения. Для этого нужно умножить базовую точку на закрытый ключ: P = k * G. В данном случае закрытый ключ равен 15, а базовая точка (2;2), поэтому открытый ключ будет равен:

P = 15 * (2;2) = (59;17

4. Зашифровать сообщение, которое нужно передать пользователю B. Для этого нужно выбрать случайное число r из диапазона от 1 до n и вычислить точки R = r * G и S = r * P + M, где M — это сообщение, которое нужно зашифровать (в данном случае M = 50). Точка R будет использоваться в качестве части открытого ключа, а точка S будет передана пользователю B.

В данном случае можно выбрать, например, r = 23. Тогда точка R будет равна:

R = 23 * (2;2) = (38;70)

А точка S будет равна:

S = 23 * (59;17) + 50 = (24;16)

5. Передать пользователю. В точку R и значение S.

6. Пользователь B может расшифровать сообщение, используя свой закрытый ключ и точку R. Для этого нужно вычислить точку T = k * R и затем вычислить значение M = S - T. В данном случае закрытый ключ пользователя B равен 15, а точка R была передана ему ранее. Поэтому можно вычислить:

T = 15 * (38;70) = (54;37)

M = (24;16) - (54;37) = (69;44)

P - открытый ключ, который используется для расшифровки сообщения, вычисляется путем умножения базовой точки на закрытый ключ: P = k * G, где k - случайное число из диапазона от 1 до порядка эллиптической кривой n, а G - базовая точка.

R - часть открытого ключа, вычисляется путем выбора случайного числа r из диапазона от 1 до n и умножения базовой точки на это число: R = r * G.

S - зашифрованное сообщение, вычисляется путем умножения открытого ключа на число r и прибавления к этому результату самого сообщения: S = r * P + M, где M — это сообщение, которое нужно зашифровать.

T - точка, которая используется для расшифровки сообщения, вычисляется путем умножения точки R на закрытый ключ пользователя B: T = k * R.

M - исходное сообщение, которое нужно расшифровать, вычисляется путем вычитания точки T из точки S: M = S - T.

Умножение точки на число на эллиптической кривой происходит следующим образом:

1. Инициализируем переменные result и temp, которые будут хранить результаты вычислений.

2. Представляем число в двоичном виде и начинаем перебирать его биты справа налево.

3. Если текущий бит равен 1, то прибавляем к result точку, которую нужно умножить, к temp.

4. Удваиваем точку temp.

5. Переходим к следующему биту и повторяем шаги 3-4 до тех пор, пока не переберем все биты числа.

6. Возвращаем результат - точку result.

Например, чтобы умножить точку G на число k = 7, мы делаем следующие действия:

1. Инициализируем переменные result и temp: result = 0, temp = G.

2. Представляем число 7 в двоичном виде: 111.

3. Текущий бит равен 1, поэтому прибавляем к result точку G к temp: temp = 2G, result = G.

4. Удваиваем точку temp: temp = 4G.

5. Текущий бит равен 1, поэтому прибавляем к result точку G к temp: temp = 5G, result = 2G.

6. Удваиваем точку temp: temp = 10G.

7. Текущий бит равен 1, поэтому прибавляем к result точку G к temp: temp = 11G, result = 3G.

8. Все биты числа перебраны, возвращаем результат - точку 3G.

Тест2

Для того чтобы зашифровать число 50 с помощью NTRUEncrypt генерируется пара ключей: публичный и секретный. Публичный ключ используется для шифрования данных, а секретный ключ - для их расшифровки. Опишем процесс генерации ключей и шифрования:

1. Генерация ключей

a) Генерация простых полиномов

Выбираются значения параметров по умолчанию N = 503, q = 2048, df = 37, df1 = 38, df2 = 38. Далее генерируются случайные полиномы f, g из множества полиномов R_q[X] со степенями меньше df, такие что (f, gcd(g, X^N - 1)) = 1 и deg(g) < df2.

b) Генерация приватного ключа

Приватный ключ представляет собой пару значений pol_f, pol_g.

c) Генерация публичного ключа

Публичный ключ состоит из полинома h = f * g^-1

2. Шифрование

a) Перевод числа 50 в полином m

m = 50 + X

b) Генерация случайного полинома r

r = rand_polynomial(df1 - 1)

c) Шифрование

Выбираем случайный полином e из R_q[X] со степенями меньшими, чем df2. Вычисляем шифртекст с помощью формулы:

c = (r*h + e + m) mod q

Шифртекст c можно передавать по открытому каналу.

Теперь давайте применим данный алгоритм для шифрования числа 50:

1. Генерация ключей

Параметры по умолчанию N = 503, q = 2048, df = 37, df1 = 38, df2 = 38.

Генерируем случайный полином f с коэффициентами в R_q[X] со степенями меньше df и такой, что (f, gcd(g,X^N - 1)) = 1.

Генерируем случайный полином g с коэффициентами в R_q[X] таким, что deg(g) < df2.

Находим полином h = f * g^(-1).

Значения pol_f = f, pol_g = g будут использованы в дальнейшем для расшифрования данных.

2. Шифрование

Для шифрования числа 50 необходимо сначала представить его в виде полинома m:

m = [50, 1] (коэффициенты полинома начинаются с младшей степени)

Выбираем случайный полином r из R_q[X] со степенями меньшими, чем df1 - 1:

r = [1463, 1404, 306, 1778, 157, 1612, 37, 1507, 937, 539, 263, 910, 1419, 341, 1991, 1574, 393, 1583, 1773, 1076, 1929, 1798, 1106, 462, 353, 1950, 1165, 1991, 1585, 970, 393, 1991, 1793, 779, 703, 1645]

Выбираем случайный полином e из R_q[X] со степенями меньшими, чем df2:

e = [68, 461, 501, 65, 300, 205, 1394, 1308

Для расшифрования данных необходимо знать приватный ключ, который состоит из значений pol_f и pol_g, которые были сгенерированы ранее при генерации ключей.

Для расшифрования шифртекста с используется формула:

m = (c * pol_f) mod q mod pol_g

Подставляя значения pol_f, pol_g и c, получаем:

m = ((r*h + e + m) * f) mod q mod g

Это уравнение нелинейно относительно m и его решение проблематично в общем случае. Однако, если число e достаточно мало, то можно пренебречь его влиянием. Получаем уравнение:

m = ((r*h + m) * f) mod q mod g

m = ((r*(f*g^(-1)) + m) * f) mod q mod g

m = ((r*f^2) + m*f) mod q mod g

Подставляем значения для r и m, которые использовались при шифровании:

m = (([1463, 1404, 306, 1778, 157, 1612, 37, 1507, 937, 539, 263, 910, 1419, 341, 1991, 1574, 393, 1583, 1773, 1076, 1929, 1798, 1106, 462, 353, 1950, 1165, 1991, 1585, 970, 393, 1991, 1793, 779, 703, 1645]*[505, 480, 650, 710, 1845, 1983, 1216, 308, 133, 225, 987, 1883, 1229, 1022, 321, 1464, 339, 113, 1155, 779, 444, 1546, 965, 912, 1024, 992, 982, 935, 1265, 327, 566, 1223, 884, 828, 214, 1059]^-1 + [50, 1]) * [1372, 57, 465, 268, 765, 1683, 1969, 548, 231, 1074, 266, 741, 91, 1898, 1952, 85, 1935, 1385, 742, 820, 1873, 925, 981, 1105, 1769, 1550, 1881, 1985, 464, 313, 1007, 1790, 1014, 416, 622, 1140]) mod 2048 mod [1720, 1]

Вычисляем значения в скобках:

[r*(f*g^(-1)) + m] * f:

([1463, 1404, 306, 1778, 157, 1612, 37, 1507, 937, 539, 263, 910, 1419, 341, 1991, 1574, 393, 1583, 1773, 1076, 1929, 1798, 1106, 462, 353, 1950, 1165, 1991, 1585, 970, 393, 1991, 1793, 779, 703, 1645]*[505, 480, 650, 710, 1845, 1983, 1216, 308, 133, 225, 987, 1883, 1229, 1022, 321, 1464, 339, 113, 1155, 779, 444, 1546, 965, 912, 1024, 992, 982, 935, 1265, 327, 566, 1223, 884, 828, 214, 1059]^-1 + [50, 1]) * [1372, 57, 465, 268, 765, 1683, 1969, 548, 231, 1074, 266, 741, 91, 1898, 1952, 85, 1935