Основные проблемы, с которыми мы столкнулись:

1. Как найти генератор мультипликативной группы кольца вычетов по модулю?

Мы решили эту проблему так: Мы брали число P такое, что P = 2\*q+1, где число q тоже простое (такие числа называются числа Софи-Жермен). Тогда в качестве A – генератора мультипликативной группы кольца вычетов по модулю P – можно взять такое число, для которого выполняется:

Aq mod(P)≠1 и 1 < A < P – 1. Такой выбор числа P также усиливает криптостойкость.

1. Как сгенерировать большое простое число?!

Мы реализовали и использовали тест Миллера-Рабина.

1. Как сгенерировать большие числа Софи-Жермен?!

Мы просто подбирали такое большое число + проверяли на простоту тестом Миллера-Рабина. Нам нужно не просто большое такое число, а БОЛЬШОЕ. После рассуждений пришли к выводу, что числа, у которого 2048 цифр, нам хватит. Это была сложная задача. Мы арендовали VPS, где на протяжении нескольких дней беспрерывной работы было найдено такое число:

1529974132788348527827399174805408017302386621701931511575289359438864766342281493847226656939125725627141747249394863474863099243840369251705285269336920030537419118914281048870741321127933505732851176271383660943673458650339420561523629024461612122336571464882418099678817960172486926041938958202441057181892626487886251210106376282614071416874515276460433105366542518258987161568280774109979255169804807744845213993173600964055046677945130837604468115720596464181198460518018926875230912344976673721645994574225344720394313027841454060896218702880268162047508452883764192446212528875845563910804327030079312293617449614586486137447460363854214082805027528204460629463363687022727692258536914382613818433760286923175142135287656473244227836100984832597405108970298476243924457372836263344464047278017705058780763987813512072104081320727193392116964664621597673471955573926472787540386138440936133219652841018812645533024121419191426921991135121391997929973284878883121375806741386634177518795102904911115375258174520842722205494704069664874457758506957495412852818218199893072047475462723962606056456511445528188433712888926719162516543025333467580143677875280570448683489655894020841568898377156068691175428375222272884601024802128198250170224347420289750260232895524960597932538363343811664814893259197654040040161088919101121269253172866214359186419711101186828805365447915064924052195330242668558012371992360256014655755105804526645309167235589052144606769117012912239232286931222313339521164328130594850370855550583771835041933662889745953541851707162563178957586514188777837633265116126261358573165331960554239608190179860176639328132215190450121788492662546098086031046452443724701416794379058448460177860241193997850460322321885422564301560748172194416791883261050892531468684271333480851466160575908723914736822104753951311105236191182875250771718423305428557786465627363437506766710040327750222629351519120185490790020894376046248688624433457045616055077339266762389646535389428231613340514341515273140182844336435219802682312547741810986993

Причем если его умножить на 2 и прибавить 1, то получится тоже просто число. Такая пара чисел и называется числами Софи-Жермен

1. Какой использовать алгоритм симметричного шифрования?

Мы решили выбрать AES-256. Реализовывать всё заново мы не стали, так как это не является темой нашего альтернативного экзамена, а взяли готовое решение, встроенное в java. Точнее мы реализовали удобный интерфейс для работы с этим AES-256, а то встроенный неудобный.

1. Как лучше хранить ключи?

Мы реализовали класс, который обо всём этом заботится. Надо просто создать объект этого класса, с заданными параметрами, или он всё сгенерирует сам. У этого объекта есть методы: зашифровать и дешифровать сообщение, подписать и верифицировать сообщение, подписать и верифицировать ключи.

1. Какую лучше использовать Хэш-функцию?

Важно выбрать такую, которые используют в криптографии. Изучив порядка 10 шт, мы выбрали SHA-512. Реализовывать мы её с нуля не стали, потому что это не наша тема альтернативного экзамена, а сделали удобный интерфейс, с помощью которого можно легко пользоваться этой функцией, встроенную в java.

1. Как лучше испытать этот алгоритм Эль-Гамаля?

Решили сделать клиент-серверное приложение. Учебную программу, которая позволяла бы настроить шифрование и включить его.

1. Где найти сервер с белым IP?

Мы арендовали VPS и развернули на нём сервер.

1. Как продемонстрировать алгоритм Эль-Гамаля без программы?

Мы решили сделать сайт. На нём можно будет поиграть с алгоритмом на маленьких числах.

1. Как превратить сообщение в числа, не большие чем P?

Мы решили перевести сообщение (т. е. набор символов) в байты, далее эти байты представляют како-то очень большое число. Число “режем” так, чтобы кол-во цифр было меньше, чем у P, причём такое разрезанное число не должно начинаться с нуля. Получается, что сообщение представляется как набор этих разрезанных чисел. Чтобы восстановить исходное сообщение, нужно просто эти разрезанные числа записать одно за другим. Тогда получится какое-то большое число. Переведём число обратно в байты. Теперь эти байты представляют исходное сообщение.

1. Как кодировать сообщения, зашифрованные алгоритмом Эль-Гамаля?

В алгоритме сообщение шифруется, затем передаётся в виде 2-х чисел. Чтобы пользователям было легче передавать ключи нужно придумать способ как из двух чисел сделать одно так, чтобы можно было восстановить их. Мы решили эту проблему следующим образом: числа *r* и *e* переводятся в байты, затем смотрим из скольки байт состоит число *r*. Обозначим это число за *a*. Теперь a вставляем перед этими числами. Теперь чтобы получить число r нужно просто считать число *a* и считать *a* байт начиная с конца *a*. Чтобы считать число e нужно посмотреть сколько байт представляет число r и начать считывать с конца *r* до конца. Представление будет выглядеть так: [*a*][*r*][*e*], где [*a*] – байты, представляющие число a, [*r*] – число *r* и [*e*] – число *е*.