Периодически в приложениях необходимо сравнивать две величины. Рассмотрим частный случай - сравнение массивов (или строк) применительно к криптографии.

В большинстве языков программирования, реализована операция (или функция) сравнения массивов (или строк) обозначаемая '==' или методом *.equals(*)

В подавляющем большинстве случаев, данная операция выполняется по следующему алгориму:

bool areEqual(int arr1[], int arr2[], int n, int m)   
{   
 // If lengths of array are not equal means   
 // array are not equal   
 if (n != m)   
 return false;   
   
 // Linearly compare elements   
 for (int i=0; i<n; i++)   
 if (arr1[i] != arr2[i])   
 return false;   
   
 // If all elements were same.   
 return true;   
}

Как видно из кода, в целях оптимизации метод завершает свою работу при первом несовпадении. Таким образом, время выполнения алгоритма зависит от подаваемых на вход величин: массивы, отличающиеся в последнем байте, будут сравниваться дольше, чем массивы, отличающиеся в первом.

Для большинства приложений, такое поведение метода сравнения является корректным и приемлемым, так как позволяет экономить до n/2 операций при сравнении случайных массивов.

Для криптографии - не всё так радужно, так как использование данного метода позволяет реализовать так называемую атаку по времени, являющуюся частным случаем атаки по побочным каналам.

Пусть имеется сервер, проверяющий HMAC, присланный пользователем. Для этого вычисляется целевой HMAC от некоторых данных, а затем сравнивается с HMAC'ом, присланным пользователем, используя стандартный метод '==', описанный выше.

Очевидно, что время ответа сервера зависит от того, насколько MAC пользователя совпадает с целевым. При этом время проверки линейно зависит от числа символов в совпавшем префиксе.

Противник может осуществить атаку - имея корректный префикс длины i-1, он перебирает i-й байт в последовательности, замеряя время ответа. Если время ответа для некоторого символа больше, чем для остальных - этот символ считается угаданным, он добавляется в префикс и перебирается i+1 символ.

## Задание

Дана REST служба с API указанным ниже.

Задача - получить от севера ответ Wellcome to SecretNet!

NB - для простоты лр и уменьшения времени перебора размер MAC был ограничен 8 байтами. Для "боевого" исполнения размер в 32 байта не является проблематичной для данного типа атак, но время атаки возрастёт линейно.

## Ход работы.

### Тестирование

<userId> = имя аккаунта на GitHub (или фамилия студента)

<challengeId> = 1

1. Проверить работоспособность контроллера с помощью метода GET <host>/api/HmacTiming
2. Получить ответ Wellcome to SecretNet!
3. Найти минимальное значение параметра d=<delay>, при котором код перестаёт работать
4. Написать лр (не удаля старый код!!!) который выполнит лр для значения d/2
5. Найти по крайней мере два алгоритма, с помощью которых можно реализовать сравнение за константное время.
6. Найти, если ли в вашем языке программирования встроенный метод, для сравнения за константное время (может быть методом у хэш функций).

### Сдача лр

шаги 1 - 3 этапа тестирования аналогично для 20 различных <challengeId>.

## Описание API

Rest запросы, в заголовке выстален Content-Type: application/json; charset=utf-8.

### Описание методов

## GET <host>/api/HmacTiming

Проверка работоспособности контроллера. Возвращает operating. Ответ не кодируется в BASE64.

|  |  |
| --- | --- |
| Параметр | Описание |
| <host> | имя хоста веб службы |

## GET <host>/api/HmacTiming/<userId>/<challengeId>/data=<data>&mac=<mac>&delay=<delay>

Проверяет аутентичность пользовательских данных. Для сравнения MAC используется сравнение, с непостоянным временем исполнения.  
Для простоты лр дополнительную задержку можно указывать в параметре <delay>

|  |  |
| --- | --- |
| Параметр | Описание |
| <host> | имя хоста веб службы |
| <userId> | идентификатор студента |
| <challengeId> | идентификатор задания |
| <data> | данные, на которых вычисляется MAC, hex строка |
| <mac> | MAC для проверки, hex строка |
| <delay> | задержка в мс на один символ в операции сравнения, принимает значения 1...1000 |