ОСОБЕННОСТИ ПРИМЕНЕНИЯ ДЛИННОЙ АРИФМЕТИКИ ДЛЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ И AZURE MOBILE SERVICES НА ПРИМЕРЕ ПРОТОКОЛА АУТЕНТИФИКАЦИИ SRP6A

Проект КИБЭВС-1505

М. О. Калмыков E-mail: kmo.azure@gmail.ru, *Е. В. Рассказов* E-mail: rev7.azure@gmail.ru
Научный руководитель: *И. В. Горбунов* E-mail: *giv@keva.tusur.ru*

Введение. На сегодняшний день, когда большая часть личных данных хранится на просторах интернета, то чтобы защитить данные от злоумышленников, необходимо использовать основные методы защиты информации, такие как криптография, аутентификация, хеширование и т.д. Данные методы защиты строятся на математике с большими числами, типа 10^{100} и больше, а так как основные типы функциональных языков программирования не позволяют реализовать математические функции, типа сложение, умножение и возведение в степень, то необходимо использовать «Длинную арифметику». Поэтому при реализации операций с большими числами, возникают некоторые сложности, связанные со спецификой языков программирования, на которых идёт реализация методов защиты информации, указанные в таблице 1 [1].

Таблица 1 – Особенности реализации длинной арифметики на основных функциональных языках программирования

Язык программирования	Порядок бит	Наличие отрицательного значения	Число 30000 в байтах	Число 15777216 в байтах
C#	LittleEndian	Есть	30-75	C0-BD-F0-00
Java	BigEndian	Есть	75-30	00-C0-BD-F0
C-Object	BigEndian	Нет	75-30	C0-BD-F0

При разработке мобильного приложения для безопасного хранения аутентификационных данных под основные мобильные платформы (Windows Phone, Android и IOS) в рамках программы группового проектного обучения «КИБЭВС – 1505», был реализован протокол аутентификации SRP6a. SRP6a – протокол аутентификации пользователей приложений [2]. Он устойчив к различным хорошо известным атакам, проводимым в сетях общего пользования. Данный протокол принадлежит к семейству надежных протоколов EKE (Encrypted Key Exchange). Протокол SRP6a рекомендуется использовать, когда требования конфиденциальности выше, чем может обеспечить современная версия HTTPS протокола TLS 1.2 [3]. Однако реализации данного протокола в виде библиотек открытого кода устарели, а под некоторые платформы и вовсе не существуют. При самостоятельной реализации SRP6a разработчики сталкиваются с рядом проблем, вызванных тем, что протокол основан на математических операциях с большими числами, прежде всего на вычисления по модулю числа N. Реализация больших чисел даже на одном языке программирования, но на разных платформах, в том числе мобильных, отличается. Этот факт становится особенно важным при самостоятельной реализации функций аутентификации в процессе разработке мобильных сервисов на платформе Azure. Самостоятельная реализация была выбрана прежде всего ввиду непозволительности использования аутентификации через сторонние сайты и социальные сети, а также ввиду отсутствия готовой реализации SRP6a в мобильных сервисах Azure [4].

SRP6 и особенности его реализации. Для представления протокола SRP6a введем обозначения:

q и $N=2q+1$ - выбираются так, что N и q взаимно простые; N должно быть достаточно большим, чтобы дискретное логарифмирование по модулю N было практически

неосуществимо (в системе длина составляет 4096 бита), N генерируется для каждой регистрации отдельное на основе значений I и s,

g — генератор мультипликативной группы Z_N,

k — параметр, получаемый на обеих сторонах, $k=H(N, g)$,

s — соль,

I— идентификатор пользователя в системе сервера (username),

p — пароль пользователя,

$H()$— криптографическая хеш-функция SHA-512,

x— секретный ключ, $x = H(s, p)$,

v — верификатор пароля на стороне сервера, $v=g^x$, u— произвольный параметр для кодирования,

a, b— секретные одноразовые числа. Вся арифметика выполняется по модулю N.

Кратко описать протокол SRP6a можно в виде диаграммы последовательности UML, представленной на рисунке 1.

Из описания протокола видно, что необходимо добиться единого представления больших чисел для математических операций и в виде последовательности байт для одинокого результата выполнения хеш-функций.

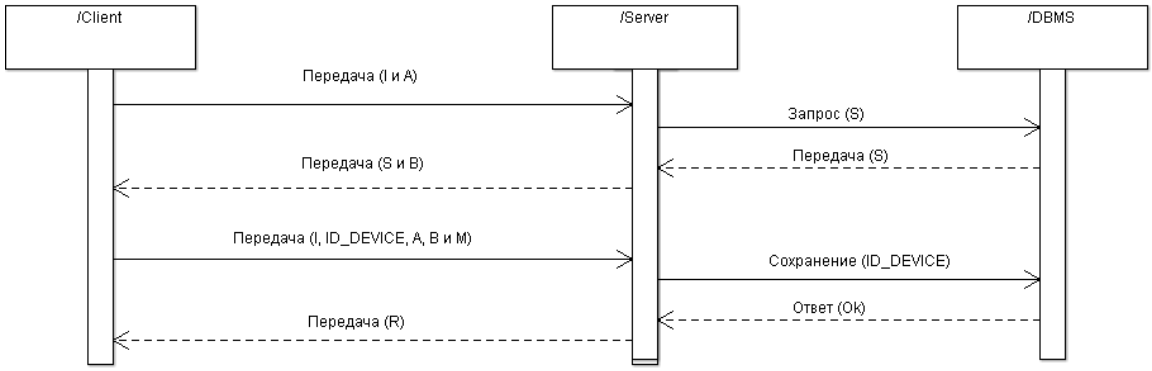

Рис. 1. Диаграмма последовательности взаимодействия при аутентификации между клиентом и облачным сервисом по протоколу SRP6a

Реализация SRP6a в языке C# в виде веб-сервиса на платформе WebApi (Azure Mobile Service) и универсального приложения для Windows и Windows Mobile (Universal Windows Apps) не вызывает особых проблем кроме необходимости постоянного контроля положительности результатов всех операций над большими числами, чего можно добиться умножением на -1 при необходимости. Большие числа реализуются стандартным типом System.Numerics.BigInteger.

Пример:

—5458216386387045526576437686524799174964636830365711715731857089285612889614585940804807703058057440951374750779968067461088802729759447810697171725429594210542646896936561921011889132569192929186294580806519215018278046593286659180855818840318019826128750692025418994546998034229807884916518211245311978531093519562042971182420964632842190001846526372314465291746632834934131607681157436352705931051758685356731551300907974071730921258033871134484140065483765727818720754317589176911488206619834283817101070588198135655120592140530675620036300573747484682776423150200826490881402822065523933859839731411605941657677924423855214520276711337526789464215156891290649859756653439559299132043967707669560443711899292740334464188303128045279095333263362421312605549621151439716570804083980972768531002347621946618922832285724471828731813245605737170410303599503430173221931113434672548872063729469468480891358750636262033813024989571822990968160016367720414347954401017508791946617636025222069636689997605335401153959345651818058697352531959078489544798423801891248205285618973014706892657515195690207514079158424506197505 8

В таком случае число умножается на -1.

При реализации этого же протокола на языках Java, JavaScript для платформ Android и HTML 5 стоит учесть, что для получения такого же с математической точки зрения числа, байты необходимо переставить задом наперед (операция реверс). Большие числа для Android реализуются нативной стандартной библиотекой `java.math.BigInteger`, а для платформы HTML Big Integer Library v.5 (http://www.leemon.com).

Пример:

Одно и тоже число в виде Big Number.

В байтах на Windows Phone:

ccb9d9f5ee … 99bbc73c6e

В байтах на Android:

6e3cc7bb99…eef5d9b9cc

Известно, что у компании Apple свои взгляды на все стандарты. Работа с большими числами также отличается от представленных выше примеров. Большие числа не только следует в байтах указывать в обратном порядке, но и обязательно требовать первым битом 0. Так, стандартное умножение на -1 не поможет, так как отрицательных значений на платформах iOS у больших чисел не подразумевается, а 1 в первом бите большого числа будет дополнением до ближайшего 2^N числа, большего того, которое требуется представить. Поэтому, с целью соблюдения единства аутентификации, для всех ранее реализованных платформ требуется обеспечить проверку на наличие 0 в первом бите большого числа путем его добавления при необходимости.

Пример:

Представления числа в байтах:

13-38-F4-52-C8-CA-FD-FE-D7-C3-4B-FD-14-DC-93-ED-8E-8B-6B-D0-0E-B7-AE-0C-70-69-2A-FE-F3-4B-62-8A-B6-C6-BA-DB-08-35-AC-7E-DE-21-AA-DA-6B-67-5D-52-47-05-44-33-DE-3C-DB-28-10-DC-FB-07-49-AE-1C-C2

Число в Windows Phone:

324133576631672761916860400818126371340829547857166265333094005075588148861454287718638569618741668555534608195156482268967942152991966723363414080564631 7

Число в iOS:

101664721636258694804054209900245824140710703420207307243926213929658825414590040996154886019794867421346857762349212281640744612820269027127995082004377 79

Большие числа реализуются различными библиотеками, например, JKBigInteger, основанным на фреймворке OpenSSL. Однако, OpenSSL из средств для разработки Xcode под iOS по умолчанию удален с версии 5 и выше. Требуется его самостоятельная компиляция из исходного кода, и добавление в проект.

Заключение. В результате работы были обнаружены и решены проблемы при реализации SRP6a на основе существующих реализаций длинной арифметики на основных языках программирования под такие мобильные платформы, как Windows Phone, Android и IOS, что позволило получить кроссплатформенную реализацию функции аутентификации на веб-сервере совместимую со всеми популярными клиентами.

Предложенное решение вошло в приложение Lockout доступное для телефонов и планшетов на Android, Iphone в рамках Google play market и Apple Store.

Список литературы

1. С.М. Окулов – «Длинная арифметика» URL: http://comp-science.narod.ru/DL-AR/okulov.htm.

2. The Stanford SRP Homepage URL: http://srp.stanford.edu/whatisit.html.

3. Taylor D., Wu T., Mavrogiannopoulos N., Perrin T., "Using the Secure Remote Password (SRP) Protocol for TLS Authentication" URL: http://www.ietf.org/rfc/rfc5054.txt

4. Gailey G., "Add authentication to your Mobile Services app" URL: https://azure.microsoft.com/en-us/documentation/articles/mobile-services-dotnet-backend-windows-universal-dotnet-get-started-users/