ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2010 Управление, вычислительная техника и информатика № 4(13)

УДК 004.056

П.А. Паутов

АУТЕНТИФИКАЦИЯ В МНОГОУРОВНЕВОЙ СИСТЕМЕ НА ОСНОВЕ КОММУТАТИВНОГО ШИФРОВАНИЯ И ДОВЕРЕННЫХ ПОДПИСЕЙ

В статье рассматривается подход к организации аутентификации в многоуровневой системе, известный как «модель доверенной подсистемы». Для данного подхода формулируются требования безопасности и приводятся два протокола аутентификации, удовлетворяющие этим требованиям. Первый протокол построен на основе коммутативного шифрования, а второй – на основе доверенных подписей. Для протокола на основе коммутативного шифрования рассматриваются несколько конкретных алгоритмов шифрования.

Ключевые слова: *многоуровневые системы*, *аутентификация в многоуровневых системах*, *коммутативное шифрование*, *доверенные подписи.*

Рассмотрим систему, состоящую из трёх взаимодействующих подсистем: клиент, внешний сервер, внутренний сервер. Клиент взаимодействует только с внешним сервером, внешний сервер взаимодействует как с клиентом, так и с внутренним сервером (внешний сервер является клиентом внутреннего сервера). Внешний сервер взаимодействует с внутренним только для обработки запросов своих клиентов. Например, по такому принципу организованы многие веб-приложения. В веб-приложениях в качестве клиента выступает браузер, в качестве внешнего сервера – веб-сервер и в качестве внутреннего сервера – сервер СУБД.

В таких многоуровневых системах обычно используется одна из следующих двух моделей организации аутентификации [1], представленных на рис. 1:

1) модель делегирования;

2) модель доверенной подсистемы.

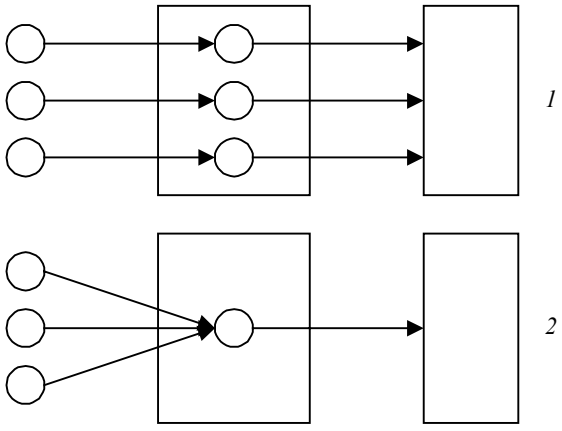

Рис. 1. Схематичное изображение моделей организации аутентификации:
1 – модель делегирования; *2* – модель доверенной подсистемы

В модели делегирования внешний сервер взаимодействует с внутренним от имени клиента, т. е. внутренний сервер содержит учётную запись для каждого клиента внешнего сервера. В модели доверенной подсистемы внешний сервер взаимодействует с внутренним от имени фиксированного набора учётных записей, т. е. одна учётная запись внутреннего сервера соответствует нескольким клиентам внешнего сервера. В данной работе будет рассматриваться модель доверенной подсистемы.

В работе автора [2] предложено несколько схем аутентификации для случая, когда между клиентом и внешним сервером и между внешним сервером и внутренним используется парольная аутентификация. В них внешний сервер применяет пароль учётной записи внутреннего сервера для того, чтобы пройти аутентификацию от имени данной учётной записи. В этом случае, если злоумышленник скомпрометирует внешний сервер так, что получит доступ на чтение к памяти внешнего сервера, то он получит пароль учётной записи внутреннего сервера, соответствующей привилегиям клиента, когда тот выполнит протокол аутентификации. После одного сеанса связи клиента со скомпрометированным внешним сервером злоумышленник получает возможность использовать внутренний сервер без помощи клиента. В данной работе предлагается несколько схем, позволяющих преодолеть этот недостаток.

1. Постановка задачи

В модели доверенной подсистемы для взаимодействия с внутренним сервером используется учётная запись, соответствующая привилегиям клиента внешнего сервера. Например, на внешнем сервере клиенты делятся на группы по привилегиям: «гости», «операторы», «администраторы». Тогда для взаимодействия с внутренним сервером можно использовать три учётные записи, соответствующие группам клиентов. Если внешний сервер сам выбирает учётную запись для взаимодействия с внутренним сервером, то в случае компрометации первого злоумышленник сможет использовать учётную запись с максимальными привилегиями. Возникает задача разработки такой схемы аутентификации, при которой внешний сервер смог бы использовать для взаимодействия с внутренним сервером только ту учётную запись, которая соответствует клиенту, и только тогда, когда клиент взаимодействует с внешним сервером. Например, если клиент, обращающийся к внешнему серверу, относится к группе «гости», то внешний сервер должен иметь возможность пройти аутентификацию перед внутренним сервером только от имени учётной записи «гость». Кроме того, внешний сервер не должен иметь возможности пройти аутентификацию перед внутренним сервером от имени учётной записи «гость» без помощи клиента, относящегося к группе «гости».

Иначе говоря, искомая схема аутентификации должна удовлетворять следующим двум условиям:

С1. При взаимодействии с клиентом внешний сервер может пройти аутентификацию перед внутренним сервером только от имени учётной записи, соответствующей данному клиенту.

С2. Внешний сервер не может пройти аутентификацию перед внутренним сервером от имени какой-либо учётной записи без помощи соответствующего клиента.

2. Схема аутентификации на основе коммутативного шифрования

Для построения предлагаемой схемы необходимы следующие криптографические примитивы:

1) E – коммутативный алгоритм шифрования, D – соответствующий алгоритм расшифрования;

2) H – хэш-функция.

Каждой учётной записи внутреннего сервера ставится в соответствие некоторый секрет S. Каждому клиенту внешнего сервера ставится в соответствие ключ K алгоритма E. На внешнем сервере для каждого клиента хранится результат шифрования $E_K(S)$, где S соответствует учётной записи внутреннего сервера для данного клиента. Алгоритм аутентификации клиента выглядит следующим образом.

1. Клиент посылает внешнему серверу своё имя.

2. Внешний сервер посылает внутреннему серверу имя учётной записи, соответствующей клиенту.

3. Внутренний сервер генерирует случайный ключ шифрования Kr алгоритма E и передаёт его внешнему серверу.

4. Внешний сервер находит запись $E_K(S)$, соответствующую данному клиенту, вычисляет $E_{Kr}(E_K(S))$ и передаёт полученный результат клиенту.

5. Клиент вычисляет $H(D_K(E_{Kr}(E_K(S)))) \equiv H(E_{Kr}(S))$ и передаёт данное значение внешнему серверу.

6. Внешний сервер передаёт полученное значение внутреннему серверу.

7. Внутренний сервер вычисляет $H(E_{Kr}(S))$ и сравнивает результат со значением, полученным от внешнего сервера. Если значения совпадают, то клиент прошёл аутентификацию перед внешним сервером, а внешний сервер перед внутренним.

Для поддержки нескольких параллельных сеансов в сообщения протокола можно добавить идентификаторы сеансов либо использовать какой-либо существующий коммуникационный протокол (обеспечивающий несколько параллельных сеансов) в качестве нижележащего протокола.

Как видно из описания, данный протокол обеспечивает выполнение условий C1, C2. Если внешний сервер попытается использовать запись $E_{K'}(S')$, не соответствующую данному клиенту, то проверка, осуществляемая на шаге 7, не выполнится, так как $H(D_K(E_{Kr}(E_{K'}(S')))) \neq H(E_{Kr}(S'))$. Для того чтобы пройти аутентификацию перед внутренним сервером без помощи клиента, внешнему серверу понадобится знание S, но это значение недоступно внешнему серверу в открытом виде.

Хэш-функция используется здесь для того, чтобы предотвратить атаку, в которой внешний сервер генерирует собственное значение Kr' и, получив от клиента $E_{Kr}(S)$, раскрывает значение S.

Используемый коммутативный алгоритм шифрования может быть как симметричным, так и асимметричным. В асимметричном варианте каждому клиенту ставится в соответствие пара ключей: открытый Ke и закрытый Kd. На внешнем сервере для каждого клиента хранятся значение $E_{Ke}(S)$ и открытый ключ клиента Ke. Наличие открытых ключей клиентов позволяет администратору системы, при смене секрета S некоторой учётной записи внутреннего сервера, обновить значения $E_{Ke}(S)$ для соответствующих клиентов. Алгоритм аутентификации аналогичен симметричному варианту (но на шаге 3 внутреннему серверу достаточно сгенерировать только открытый ключ).

3. Применимые коммутативные алгоритмы шифрования

3.1. Сложение по модулю 2

Сложение по модулю 2 можно рассматривать как симметричный коммутативный алгоритм шифрования. Функции шифрования и расшифрования в данном случае будут совпадать и иметь вид $E_K(X) = D_K(X) = X \oplus K$, где $\oplus$ – побитовое сложение чисел по модулю 2 (X и K рассматриваются как булевы векторы одинаковой длины n). Злоумышленник, скомпрометировавший внешний сервер, получит доступ к значениям вида $E_K(S)$ для всех клиентов и для всех учётных записей внутреннего сервера. Пусть внешний сервер имеет m клиентов и использует одну учётную запись внутреннего сервера. Тогда для получения доступа к внутреннему серверу злоумышленник должен будет решить следующую систему уравнений:

$$\begin{cases} K_1 \oplus S = e_1, \\ K_2 \oplus S = e_2, \\ \dots \\ K_m \oplus S = e_m, \end{cases}$$

где K_1, K_2, ..., K_m (ключи клиентов) и S (секрет внутреннего сервера) являются неизвестными, а e_1, e_2, ..., e_m – известные значения, хранимые на внешнем сервере. В данной системе S является свободной переменной и, следовательно, система будет иметь 2^n решений, так как S – булев вектор длины n. Таким образом, злоумышленнику придётся произвести полный перебор всех возможных значений S.

3.2. Возведение в степень по модулю простого числа

В качестве симметричного коммутативного алгоритма шифрования можно выбрать функцию возведения в степень по модулю большого простого числа. Пусть p – большое простое число; p является общеизвестным параметром системы. В качестве ключа выбирается число K ($1 \le K \le p–2$), взаимно простое с $p–1$. Тогда функции шифрования и расшифрования будут выглядеть следующим образом: $E_K(X) = X^K \bmod p$, $D_K(X) = X^{K^{-1}} \bmod p$. Данный алгоритм известен в литературе как шифр Полига – Хеллмана [3]. В качестве коммутативного шифра он используется в [4].

3.3. RSA

Как видно из описания предлагаемого протокола, на внешнем и внутреннем серверах необходимо лишь выполнить операцию шифрования секрета S на некотором случайном ключе Kr. Так как операции расшифрования не требуется, то для вычислений на внутреннем и внешнем серверах можно использовать некоторую ключевую хэш-функцию H'_K, коммутативную с алгоритмом E_K (т.е. $H'_{K1}(E_{K2}(X)) = E_{K2}(H'_{K1}(X))$). С учётом сказанного предлагаемый протокол будет выглядеть следующим образом:

1. Клиент посылает внешнему серверу своё имя.

2. Внешний сервер посылает внутреннему серверу имя учётной записи, соответствующей клиенту.

3. Внутренний сервер генерирует случайный ключ Kr хэш-функции H' и передаёт его внешнему серверу.

4. Внешний сервер находит запись $E_K(S)$, соответствующую данному клиенту, вычисляет $H'_{Kr}(E_K(S))$ и передаёт полученный результат клиенту.

5. Клиент вычисляет $H(D_K(H'_{Kr}(E_K(S)))) \equiv H(H'_{Kr}(S))$ и передаёт данное значение внешнему серверу.

6. Внешний сервер передаёт полученное значение внутреннему серверу.

7. Внутренний сервер вычисляет $H(H'_{Kr}(S))$ и сравнивает результат со значением, полученным от внешнего сервера. Если значения совпадают, то клиент прошёл аутентификацию перед внешним сервером, а внешний сервер – перед внутренним.

В качестве асимметричного варианта алгоритма E можно использовать шифросистему RSA с модулем n, открытой экспонентой e и закрытой экспонентой d. Каждому клиенту ставится в соответствие пара ключей ($Ke = (e, n)$, $Kd = d$). На внешнем сервере для каждого клиента хранится пара $(E_{Ke}(S), Ke) = (S^e \bmod n, (e, n))$ для соответствующего S. Функцию H' можно выбрать как $H'_K(X) = X^K \bmod n$. Тогда предлагаемый протокол будет выглядеть следующим образом:

1. Клиент посылает внешнему серверу своё имя.

2. Внешний сервер посылает внутреннему серверу имя учётной записи и модуль RSA n, соответствующие клиенту.

3. Внутренний сервер генерирует случайное число r и передаёт его внешнему серверу.

4. Внешний сервер находит запись $S^e \bmod n$, соответствующую данному клиенту, вычисляет $S^{er} \bmod n$ и передаёт полученный результат клиенту.

5. Клиент вычисляет $H(S^{erd} \bmod n) \equiv H(S^r \bmod n)$ и передаёт данное значение внешнему серверу.

6. Внешний сервер передаёт полученное значение внутреннему серверу.

7. Внутренний сервер вычисляет $H(S^r \bmod n)$ и сравнивает результат со значением, полученным от внешнего сервера. Если значения совпадают, то клиент прошёл аутентификацию перед внешним сервером, а внешний сервер – перед внутренним.

3.4. ElGamal

Пусть p – большое простое число, g – порождающий элемент группы Z_p^*; p и g являются общеизвестным параметрами системы. Каждому пользователю системы ставится в соответствие пара ключей (Ke, Kd), где Kd – случайное число, такое, что $1 \le Kd \le p\text{-}2$, а $Ke = g^{Kd}$. Каждой учётной записи внутреннего сервера ставится в соответствие некоторый секрет S. На внешнем сервере для каждого клиента хранится результат шифрования $E_{Ke}(S) = (g^k \bmod p, S(Ke)^k \bmod p)$, где S соответствует учётной записи внутреннего сервера для данного клиента, а k – случайное число, такое, что $1 \le k \le p\text{-}2$.

Тогда предлагаемый протокол будет выглядеть следующим образом:

1. Клиент посылает внешнему серверу своё имя.

2. Внешний сервер посылает внутреннему серверу имя учётной записи, соответствующей клиенту.

3. Внутренний сервер генерирует случайное число r, такое, что $1 \le r \le p\text{-}2$, и передаёт его внешнему серверу.

4. Внешний сервер находит запись $(g^k \bmod p, S(Ke)^k \bmod p)$, соответствующую данному клиенту, вычисляет $(g^k \bmod p, S(Ke)^k g^r \bmod p)$ и передаёт полученный результат клиенту.

5. Клиент вычисляет $H(g^{-kKd}S(Ke)^k g^r \bmod p) \equiv H(Sg^r \bmod p)$ и передаёт данное значение внешнему серверу.

6. Внешний сервер передаёт полученное значение внутреннему серверу.

7. Внутренний сервер вычисляет $H(Sg^r \bmod p)$ и сравнивает результат со значением, полученным от внешнего сервера. Если значения совпадают, то клиент прошёл аутентификацию перед внешним сервером, а внешний сервер – перед внутренним.

Здесь алгоритм ElGamal применяется в полном виде только для шифрования секретов S учётных записей внутреннего сервера. На 4-м шаге внешний сервер выполняет только часть алгоритма ElGamal – шифрование с помощью эфемерного ключа g^r. В качестве коммутативного шифра ElGamal также используется в работе [5].

3.5. Алгоритм с открытым ключом и сложением по модулю 2

На основе конструкции, предложенной в работе [6], можно реализовать асимметричный вариант предлагаемого протокола, используя некоторую схему с открытым ключом (E', D') и операцию $\oplus$. Каждому клиенту ставится в соответствие пара ключей (Ke, Kd). На внешнем сервере для каждого клиента хранится тройка ($S \oplus k$, $E'_{Ke}(k)$, Ke), где S соответствует учётной записи внутреннего сервера, а k – случайное число. Тогда предлагаемый протокол будет выглядеть следующим образом:

1. Клиент посылает внешнему серверу своё имя.

2. Внешний сервер посылает внутреннему серверу имя учётной записи, соответствующей клиенту.

3. Внутренний сервер генерирует случайное число r и передаёт его внешнему серверу.

4. Внешний сервер находит запись ($S \oplus k$, $E'_{Ke}(k)$), соответствующую данному клиенту, вычисляет ($S \oplus k \oplus r$, $E'_{Ke}(k)$) и передаёт полученный результат клиенту.

5. Клиент вычисляет $H(S \oplus k \oplus r \oplus D'_{Kd}(E'_{Ke}(k))) \equiv H(S \oplus r)$ и передаёт данное значение внешнему серверу.

6. Внешний сервер передаёт полученное значение внутреннему серверу.

7. Внутренний сервер вычисляет $H(S \oplus r)$ и сравнивает результат со значением, полученным от внешнего сервера. Если значения совпадают, то клиент прошёл аутентификацию перед внешним сервером, а внешний сервер – перед внутренним.

4. Схема на основе доверенных подписей

Механизм доверенных подписей (proxy signatures) позволяет одному пользователю (доверителю) делегировать возможность подписывать сообщения другому пользователю (доверенному подписчику). Доверенная подпись ставится доверенным подписчиком от имени доверителя. Концепция доверенных подписей была представлена в работе [7].

Далее будем использовать следующие обозначения: Xa, Ya – закрытый и открытый ключи участника a; $PS_{a,b}(m)$ – доверенная подпись сообщения m, поставленная пользователем b от имени a; c – клиент системы; $f(c)$ – учётная запись внутреннего сервера, соответствующая клиенту c. Каждому клиенту системы и каждой учётной записи внутреннего сервера ставится в соответствие пара ключей.

Администратор системы делегирует каждому клиенту системы право подписи от имени соответствующей учётной записи внутреннего сервера. Для делегации права подписи необходимо выполнить некоторый протокол, который определяется выбранной схемой доверенной подписи. Данный протокол должен быть выполнен перед началом работы предлагаемого алгоритма. Алгоритм аутентификации будет выглядеть следующим образом.

1. Клиент посылает внешнему серверу своё имя.

2. Внешний сервер посылает внутреннему серверу имя учётной записи, соответствующей клиенту.

3. Внутренний сервер генерирует случайное число r и передаёт его внешнему серверу.

4. Внешний сервер передаёт число r клиенту.

5. Клиент подписывает число r, т. е. вычисляет $PS_{f(c),c}(r)$ и передаёт это значение внешнему серверу.

6. Внешний сервер передаёт полученное значение внутреннему серверу.

7. Внутренний сервер проверяет полученную подпись, и, если подпись верна, то внешний сервер проходит аутентификацию перед внутренним.

На 7-м шаге внутреннему серверу достаточно убедиться, что подпись поставлена от имени учётной записи, переданной на 2-м шаге. Идентификация доверенного подписчика (клиента) перед внутреннем сервером в данном протоколе не требуется. Условия С1, С2 выполняются благодаря свойствам доверенной подписи. Корректную подпись $PS_{f(c),c}(r)$ может вычислить только клиент c, которому было делегировано право подписи от имени $f(c)$. Данный протокол может быть выполнен клиентом без посредничества внешнего сервера. Если такое свойство не желательно, то протокол можно изменить следующим образом.

Каждой учётной записи внутреннего сервера помимо пары ключей ставится в соответствие некоторый секрет S, известный обоим серверам и неизвестный клиентам. На 4-м шаге внешний сервер передаёт клиенту $E_S(r)$ вместо r. И далее в протоколе везде вместо r используется $E_S(r)$. Таким образом, клиент не сможет пройти аутентификацию перед внутренним сервером без посредничества внешнего, так как клиенту не известно значение S.

Недостатком данной схемы является то, что при смене ключей, соответствующих учётным записям внутреннего сервера, придётся заново выполнить протокол делегирования права подписи с каждым клиентом. В некоторых схемах доверенной подписи протокол делегирования права подписи состоит из передачи одного сообщения $W_{a,b}$, от доверителя a доверенному подписчику b. Такие схемы доверенной подписи позволяют избавиться от описанного недостатка. Для каждого клиента системы c на внешнем сервере будет храниться значение $E_{Yc}(W_{f(c),c})$, где E – асимметричный алгоритм шифрования с тем же пространством ключей, что и используемая схема доверенной подписи. Это значение может быть вычислено администратором без помощи клиента, т.е. администратор выполняет лишь часть протокола делегирования. Тогда предлагаемый алгоритм аутентификации будет выглядеть следующим образом:

1. Клиент посылает внешнему серверу своё имя.

2. Внешний сервер посылает внутреннему серверу имя учётной записи, соответствующей клиенту.

3. Внутренний сервер генерирует случайное число r и передаёт его внешнему серверу.

4. Внешний сервер передаёт клиенту пару $(r, E_{Yc}(W_{f(c),c}))$.

5. Клиент расшифровывает на своём закрытом ключе Xc значение $W_{f(c),c}$ и с помощью него вычисляет подпись $PS_{f(c),c}(r)$, которая затем передается внешнему серверу.

6. Внешний сервер передаёт полученное значение внутреннему серверу.

7. Внутренний сервер проверяет полученную подпись, и, если подпись верна, то внешний сервер проходит аутентификацию перед внутренним.

Этот вариант алгоритма позволяет сменить ключи учётной записи внутреннего сервера или изменить соответствие между клиентами и учётными записями внутреннего сервера без участия клиентов. В качестве схемы доверенной подписи в данном протоколе можно использовать схему MUO [7], а в качестве E – алгоритм ElGamal.

Заключение

Рассмотренные в данной работе протоколы аутентификации предоставляют более сильные гарантии по сравнению со схемой, описанной в работе [2]. При использовании данных протоколов, даже после сеанса связи клиента со скомпрометированным внешним сервером, злоумышленник не сможет использовать внутренний сервер без помощи клиента. Однако для внедрения любого из данных протоколов потребуется добавить его поддержку на все уровни приложения, в то время как описанная в работе [2] схема опирается на широко распространённую парольную аутентификацию. В зависимости от требований к конкретной системе можно использовать тот или иной подход к организации аутентификации.

ЛИТЕРАТУРА

1. *Chong F.* Trusted subsystem design // MSDN. 2006. URL: http://msdn.microsoft.com/en-us/library/aa905320.aspx
2. *Паутов П.А.* Проблема аутентификации в многоуровневых приложениях // Прикладная дискретная математика. 2008. № 2. С. 87–90.
3. *Schneier B.* Applied cryptography: protocols, algorithms, and source code in C, Second Edition. Wiley, 1996. 785 p.
4. *Feng Bao*, *Robert H. Deng*, *Peirong Feng.* An efficient and practical scheme for privacy protection in the e-commerce of digital goods // ICISC 2000, LNCS 2015. 2001. P. 162-170.
5. *Weiliang Zhao*, *Vijay Varadharajan*, *Yi Mu.* A secure mental poker protocol over the Internet // ACSW Frontiers. 2003. P. 105–109.
6. *Weis S.* New foundations for efficient authentication, commutative cryptography, and private disjointness testing. MIT, 2006.
7. *Mambo M.M.*, *Usuda K.*, *Okmamoto E*. Proxy signatures: delegation of the power to sign message // IEICE Transaction Functional E79-A. 1996. V. 9. P. 1338–1354.

Паутов Павел Александрович
Томский государственный университет
E-mail: __Pavel__@mail.ru

Поступила в редакцию 16 сентября 2010 г.