Московский физико-технический институт
(государственный университет)
Кафедра Радиотехники

«Уязвимости алгоритмов формирования ЭЦП: DSA, ECDSA»

(Vulnerabilities of the formation of digital signature algorithms: DSA, ECDSA)

Эссе по курсу "Защита информации"
Костин Михаил 312 гр.

Долгопрудный, 2007.

Содержание

1. Описание алгоритмов формирования ЭЦП на базе DSA, ECDSA

1.1. DSA

Генерирование ключа:

1. Выбирается просто число с битовой длинной 160.
2. Выбирается простое число $p=qz+1$ с битовой длинной L, $512 \leq L \leq 1024$, L делится на 64.
3. Выбирается h, $1<h<p-1$, такое, что $g=h^z \bmod p > 1$.
4. Выбирается произвольное x, $0<x<q$.
5. $y=g^x \bmod p$.
6. Открытый ключ: (p, q, g, y). Секретный ключ: x.

Формирование подписи:

1. Выбирается произвольное k, $0<k<q$.
2. $r=(g^k \bmod p) \bmod q$.
3. $s=(k^{-1}(\text{SHA-1}(m)+xr)) \bmod q$
4. Если r=0 или s=0, вернуться к шагу 1.
5. Подпись (r, s)

Проверка подписи:

1. Если не выполнено $0<r<q$, $0<s<q$, подпись не верна.
2. $w=s^{-1} \bmod q$
3. $u_1=(\text{SHA-1}(m)w) \bmod q$
4. $u_2=(rw) \bmod q$
5. $v=((g^{u1}y^{u2}) \bmod p) \bmod q$
6. Подпись верна, если v=r

1.2. ECDSA

Генерирование ключа:

Начальные параметры эллиптической кривой:

1. p – модуль эллиптической кривой.
2. a, b – параметры эллиптической кривой.
3. q – порядок циклической подгруппы точек эллиптической кривой (простое число)
4. G – точка эллиптической кривой.
5. n – порядок группы точек эллиптической кривой (q – делитель n)
6. Секретный ключ: $0<d_A<q$
7. Открытый ключ: $Q_A=d_AG$

Формирование подписи:

1. e = SHA-1(m).
2. Выбирается произвольное k, $0<k<q$.
3. $r=x_1 \bmod q$, $(x_1, y_1)=kG$. Если r=0, возвращаемся к шагу 2.
4. $s=k^{-1}(e+rd_A) \bmod q$. Если s=0, возвращаемся к шагу 2.
5. Подпись (r, s).

Проверка подписи:

1. Если не выполнено $0<r<q$, $0<s<q$, подпись не верна.
2. e=SHA-1(m).
3. $w=s^{-1} \bmod q$
4. $u_1=ew \bmod q$
5. $u_2=rw \bmod q$
6. $(x_1, y_1)=u_1G+u_2Q_A$.
7. Подпись верна, если $x_1=r \bmod q$.

2. Уязвимость алгоритма DSA

Описание уязвимости:

Предположим, что в качестве k на первом шаге, мы получили «1», соответственно r=g
Тогда получаем: s=(H(m)+xr) mod q. Таким образом для любого другого сообщения мы можем вычислить его подпись не зная секретного ключа, а именно:

1. Проверяем подписи на совпадение r и g.
2. Запоминаем подпись данного сообщения (r_1, s_1).
3. Формируем хэш от полученного сообщения (H_1) и от своего сообщения (H_2).
4. Подписываем наше сообщение (s_2, r_2): $s_2 = s_1+H_2-H_1$. $r_2=r_1$.

Вероятность получения сообщения с r=g очень мала $(1/q)\sim10^{-50}$. Но, если, например, банк имеет общее g для всех клиентов, то вероятность совпадения r и g резко возрастает.

В данном случае, более вероятна другая ситуация: умышленное формирование подписи с k=1.

Пример:

Злоумышленник, работающий в банке, оформляет документ и отсылает его с подписью r=g. После этого можно отсылать с другого адреса другие документы с сформированной подписью без знания секретного ключа.

Проблема, естественно, будет заключаться в неудачной генерации подписи, которая легко подделывается.

Решение:

Данная уязвимость легко исправляется двумя способами:

1. Ввести более строгие ограничения на k: 1<k<q.
2. Использовать различные p, q, a для различных операций.

3. Уязвимость алгоритма ECDSA

Описание ошибки:

В американском стандарте ЭЦП известном как ECDSA (Elliptic Curve DSA) существует серьёзная ошибка, позволяющая выбрать такое значение секретного ключа, чтобы можно было получить одинаковые подписи для разных документов. Она доступна практически любому пользователю ЭЦП, а не только самим разработчиками программ ЭЦП.

Эта ошибка вызвана равенством x-координат противоположных точек эллиптической кривой

$$x_G = x_{-G}$$

Легко увидеть, что из qG=0, следует:

$$(q-1)G = -G$$

Отсюда:

$$r_1 = x_{kG} = r_2 = x_{(q-1)kG} = r$$

где k – случайный ключ (для простоты принимаемый за 1, т.к. вывод формул для случая k>1 аналогичен).

Опишем алгорим получения одинаковых подписей для сообщений m_1 и m_2:

1. Пусть $k_1=1$, $k_2=q-1$, тогда $r_1=r_2=r=x_G$
2. $e_1=SHA\text{-}1(m_1)$, $e_2=SHA\text{-}1(m_2)$
3. $s=k^{-1}(e+dr) \bmod q$

В нашем случае:

4. $s_1=k_1^{-1}(e_1+dr) \bmod q$
5. $s_2=k_2^{-1}(e_2+dr) \bmod q$, где $k_1^{-1} \bmod q=1$, $k_2^{-1} \bmod q = q-1$;

   Очевидно, что $s_2=s_1=s$ если $(e_1+dr) = (n-1)(e_2+dr) \pmod q$
6. $2dr=(q-1)(e_2+e_1) \pmod q$

Отсюда легко находится d:

7. $d=(2r)^{-1}(q-1)(e_1+e_2) \bmod q$

Таким образом, мы получаем абсолютно одинаковые подписи (s,r) для разных сообщений.

Решение:

1. Для исправления ошибки необходимо всего лишь обеспечить соответсвующую генерацию d. Выбирать случайное значение t и генерировать d=SHA-1(t). При данной реализации выбрать любое значение d не получится.
2. Передавать в качестве подписи не (s, r), а (s, R), где R=kG.

4. Использованные ресурсы

1. A.B. Кобец «Substitution of document signed under new American format ECDSA» 29.10.02. http://bugtraq.ru/cgi-bin/forum.mcgi?type=sb&b=15&m=62564
2. Wikipedia «Digital Signature Algorithm» http://en.wikipedia.org/wiki/Digital_Signature_Algorithm
3. Wikipedia «Elliptic Curve DSA» http://en.wikipedia.org/wiki/Elliptic_Curve_DSA
4. A.B. Кобец «Механизм мошеничества с цифровой подписью на базе российского стандарта ГОСТ Р 34.10-94» 05.04.02 http://bugtraq.ru/library/crypto/gost.html