*А. В. Ковалев, М. А. Иванов*

# НЕОБНАРУЖИВАЕМОЕ ВО СТАНОВЛЕНИЕ СЕКРЕТНОГО КЛЮЧА ДЛЯ АЛГОРИТМА ЦИФРОВОЙ ЭЛЕКТРОННОЙ ПОДПИСИ ECDSA

## Введение

Пусть существует устройство (или программа), подписывающее произвольные подаваемые на вход данные с использованием заданного ключа по алгоритму ECDSA.

Возникает вопрос: может ли производитель внести в устройство такую «закладку», которая позволит ему восстанавливать секретные ключи пользователей? В работе [2] предлагается ряд методов для криптосистемы RSA. В статье предлагаются аналогичные методы для ECDSA — стандарта цифровой подписи на основе эллиптических кривых.

## 1. Краткое описание ECDSA

Для формирования подписи задаются характеристики $(q, F)$ поля $GF(2^q)$, в котором ведутся вычисления, параметры $(a, b)$ используемой эллиптической кривой и ее порядок $n$. Должна быть известна базовая точка кривой $G$.

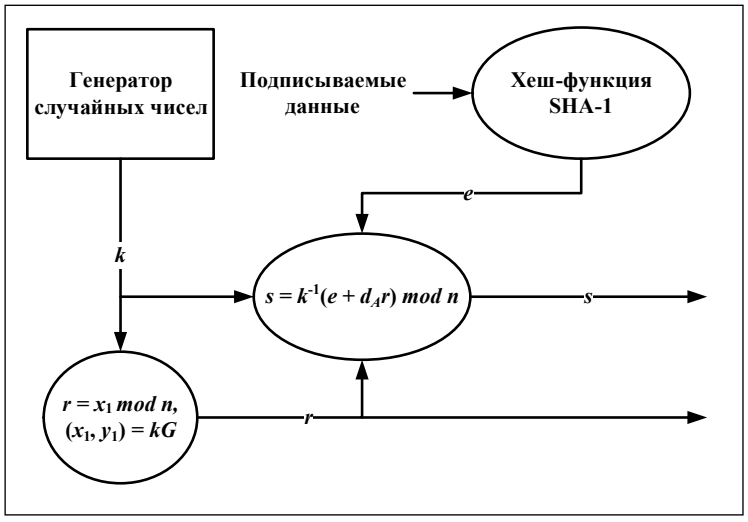


*Рис. 1. Схема формирования подписи ECDSA*

Подписывающая сторона $A$ должна обладать секретным ключом $d_A$ (случайным числом в диапазоне $[1, n-1]$), а также открытым ключом $Q_A = d_A G$. Алгоритм формирования подписи по стандарту ECDSA выглядит следующим образом:

1. Вычисляем хеш-образ сообщения $m$: $e = H(m)$, где $H(x)$ — криптографическая хеш-функция SHA-1.

2. Выбираем случайное число $k$ из диапазона $[1, n-1]$.

3. Вычисляем $r = x_1 (mod\ n)$, где $(x_1, y_1) = kG$. Если $r = 0$, возвращаемся к шагу 2.

4. Вычисляем $s = k^{-1}(e + d_A r)(mod\ n)$. Если $s = 0$, возвращаемся к шагу 2.

5. Подписью считается пара $(r, s)$.

Алгоритм проверки подписи с использованием открытого ключа $Q_A$ имеет вид:

1. Проверить, что $r$ и $s$ — целые числа из диапазона $[1, n-1]$. Если это не так, то подпись отвергается.

2. Вычислим $e = H(m)$, где $H(x)$ — криптографическая хеш-функция SHA-1.

3. Вычислим $w = s^{-1}(mod\ n)$.

4. Вычислим $u_1 = ew(mod\ n)$, $u_2 = rw(mod\ n)$.

5. Вычислим $(x_1,y_1) = u_1G + u_2Q_A$.

Подпись принимается тогда и только тогда, когда $x_1 = r(mod\ n)$.

## 2. Атака на генератор случайных чисел

Очевидно, что в случае, когда злоумышленник знает число $k$, сформированное при генерации подписи, он может сразу же вычислить секретный ключ:

$d_A = (sk - e)r^{-1}\ (mod\ n)$.

Простейший способ восстановления ключа производителем криптографического устройства заключается в подмене генератора случайных чисел генератором псевдослучайных чисел (ПСЧ), например, построенным на основе блочного шифра AES в режиме формирования гаммы с некоторым фиксированным начальным значением и фиксированным ключом.

В этом случае для восстановления секретного ключа владельца устройства (например, смарт-карты) злоумышленнику потребуется повторять формирование гаммы до тех пор, пока не будет получено число $k_i'$, удовлетворяющее условию

$r' = r$, где $r' = x_1'\ mod\ n$, а $(x_1', y_1') = k_i'G$.

Поскольку качественная псевдослучайная последовательность практически неотличима от истинно случайной, ни владелец устройства, ни сторонний наблюдатель не сможет выявить наличие «закладки».

Недостатком такой схемы является то, что, исследовав единственный экземпляр смарт-карты, можно определить начальное состояние и ключ генератора гаммы, что позволит узнать секретный ключ для любого экземпляра этого устройства.

## 3. Организация скрытого канала

Можно предложить различные пути устранения отмеченного недостатка схемы. Например, в каждый экземпляр устройства при производстве может «прошиваться» индивидуальный ключ, с помощью которого будет генерироваться ПСЧ. Однако в такой схеме для восстановления секретного ключа необходимо знать, какой именно экземпляр устройства использован для генерации подписи. Если нет точной информации на этот счет, то поиск ключа может занять значительное время.

Таким образом, мы подходим к тому, что для осуществления подобной атаки необходимо передать некоторую секретную информацию от устройства либо знать ее заранее (как в случае с заведомо известными параметрами генератора ПСЧ).

Логичным путем является сокрытие передаваемой информации в формируемых подписях. Очевидно, что необходимо исключить возможность выявления секретной информации. Для этого можно использовать асимметричную криптосистему, что позволит хранить в устройстве открытый ключ зашифрования, в результате, даже исчерпывающий анализ внутреннего устройства системы не позволит получить открытый текст.

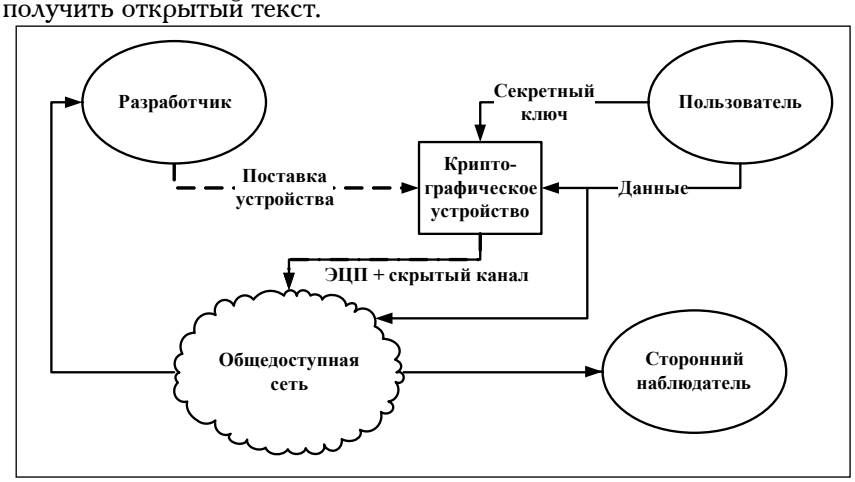


*Рис. 2. Модель протокола ЭЦП со скрытым каналом*

Сам скрытый канал проще всего организовать следующим образом. Пусть зашифрованный ключ, который следует незаметно передать, имеет размер 320 бит. Если каждая подпись будет содержать 2 бита шифротекста, то для передачи всего сообщения потребуется 160 подписей, сделанных последовательно на одном и том же устройстве. Однако даже если будет получена не вся информация, недостающую часть ключа можно будет восстановить путем перебора.

Обратим внимание на следующие этапы формирования подписи:

2. Выбираем случайное число $k$ из диапазона $[1, n-1]$.

3. Вычисляем $r = x_1 (mod\ n)$, где $(x_1, y_1) = kG$. Если $r = 0$, возвращаемся к шагу 2.

Причем $r$ является частью подписи. Несколько бит полученного шифротекста можно передать, например, в младших битах $r$. Для этого следует формировать новое случайное число $k$ до тех пор, пока младшие биты $r$ не будут совпадать с требуемыми. В среднем это приведет к увеличению времени выполнения двух названных операций в $2^t$ раз, где $t$ — число передаваемых бит.

Данный способ построения скрытого канала предельно прост, однако обладает существенным недостатком: статистические свойства двух выбранных разрядов подписи полностью определяются скрываемой информацией, тогда как в исходном алгоритме ЭЦП они зависят от характеристик генератора случайных чисел и от свойств операции умножения точки эллиптической кривой на число. Иначе говоря, появляется потенциальная возможность выявить наличие «закладки». Как этого избежать? Нужно каким-то образом распределить эти два бита передаваемой информации по всей подписи. Достаточно фиксировать два разряда не в самой подписи, а, например, в ее MAC-коде. Иначе говоря, алгоритм формирования подписи преобразуется следующим образом:

1. Вычисляем хеш-образ сообщения $m$: $e = H(m)$.

2. Выбираем случайное число $k$ из диапазона $[1, n-1]$.

3. Вычисляем $r = x_1 (mod\ n)$, где $(x_1, y_1) = kG$. Если $r = 0$, возвращаемся к шагу 2.

4. Вычисляем $s = k^{-1}(e + d_A r)(mod\ n)$. Если $s = 0$, возвращаемся к шагу 2.

5. Подписью считается пара $(r, s)$.

6. Вычисляем MAC-код подписи: $M = MAC(r|s)$.

7. Если два младших разряда $M$ совпадают с битами, которые требуется передать, завершаем алгоритм, иначе возвращаемся к шагу 2.

Для формирования кода MAC подойдет любая качественная хеш-функция с секретным вектором инициализации либо алгоритм на основе блочного шифра.

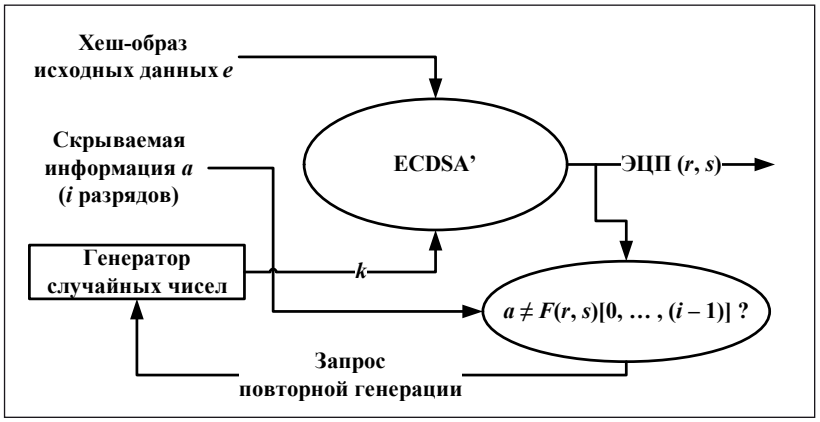


*Рис. 3. Схема скрытого канала*

Подобный подход позволит расширить скрытый канал. Количество информации, передаваемой в одной подписи, будет ограничено лишь допустимым снижением производительности. При этом защищенность канала значительно возрастает.

### 4. Использование вероятностного шифрования

При наличии скрытого канала очевидным решением является передача самого секретного ключа, однако в этом случае при попытке повторной передачи того же ключа стороннему наблюдателю станет заметна цикличность в младших разрядах *r*, что, конечно, недопустимо.

Для решения этой проблемы целесообразно применить так называемое вероятностное шифрование [3], позволяющее получать в результате зашифрования одного и того же сообщения на одном и том же ключе большое множество различных шифротекстов.

В простейшем варианте это достигается за счет зашифрования конкатенации шифротекста и достаточно длинной случайной строки. При расшифровании случайное число просто отбрасывается.

Однако такое решение является не самым изящным. Гораздо выгоднее использовать асимметричную криптосистему, изначально обладающую вероятностными свойствами. Таким образом, вероятностное шифрование позволяет передавать один и тот же секретный ключ произвольное число раз без риска выявления атаки.

### 5. Комбинирование генератора ПСЧ и вероятностного шифрования

Последняя схема имеет один недостаток. Если устройство формирует подписи с большим количеством различных секретных ключей, то для восстановления всех ключей потребуется передать каждый из них в отдельности. Вариант с использованием генератора ПСЧ свободен от этого недостатка. Попытаемся объединить достоинства двух техник.

Можно пойти следующим путем. При первом запуске устройства сформировать случайные параметры генератора ПСЧ и в дальнейшем не менять их. Эти параметры следует зашифровать при помощи открытого ключа, хранимого в устройстве (с использованием вероятностного шифрования). Далее этот шифротекст передается при помощи скрытого канала. Как только он передан полностью, формируется новый случайный параметр вероятностного шифрования и с его помощью — новый шифротекст.

Таким образом, злоумышленнику, владеющему ключом расшифрования, достаточно получить достаточное количество подписей для прочтения одного шифротекста, и после расшифрования он получит возможность узнать *любой* секретный ключ, который будет использован для формирования подписи на этом устройстве.

### 6. Использование гибридного шифрования на основе эллиптических кривых

Схемы гибридного шифрования позволяют объединить скорость симметричных криптосистем со свойствами асимметричных.

Рассмотрим использование несколько упрощенной криптосистемы ECIES [4]. Алгоритм зашифрования сообщения $m$ с использованием открытого ключа получателя (полученного как $Q_B = d_B G$, где $d_B$ — секретный ключ получателя) и тех же общих характеристик системы ($q$, $F$, $a$, $b$, $G$, $n$), что и в разделе 2, будет выглядеть так:

1. Формируем пару сеансовых ключей ($k$, $R$), где $k$ — случайное число, а $R = kG$ — точка на эллиптической кривой.

2. Вычисляем $\Phi = k\ Q_B$.

3. Вычисляем $\sigma = x_1 \bmod n$, где $(x_1, y_1) = \Phi$; если $\sigma = 0$, возвращаемся к шагу 1.

4. Вычисляем ключ симметричного шифрования $K_e = H(\sigma)$, где $H(x)$ — криптографическая хеш-функция, например SHA-1.

5. Формируем шифротекст $c = E(K_e, m)$, где E — функция зашифрования качественного симметричного криптоалгоритма, например AES.

6. Передаем получателю пару ($R$, $c$).

Последовательность расшифрования имеет вид:

1. Вычисляем $\Phi = d_B R$.
2. Вычисляем $\sigma = x_1 \bmod n$, где $(x_1, y_1) = \Phi$.
3. Получаем ключ симметричного шифрования $K_e = H(\sigma)$.
4. Расшифровываем сообщение $m = D(K_e, c)$, где $D$ — функция расшифрования.

В данном применении подобный алгоритм позволяет, во-первых, использовать для асимметричного шифрования эллиптические кривые, что потребует небольших аппаратных затрат (например, по сравнению с RSA), во-вторых, этот криптоалгоритм обладает вероятностным свойством, в-третьих, можно будет использовать существующие библиотеки для работы с эллиптическими кривыми, а в-четвертых, гибридные криптосистемы во многих случаях позволяют получить шифротекст меньшей длины, чем при использовании асимметричных алгоритмов. Также нужно отметить, что эта схема считается одной из самых стойких гибридных криптосистем на основе эллиптических кривых.

Если применить этот алгоритм в схеме, описанной в разделе 5, то можно добиться интересного эффекта: для передачи первого секретного ключа потребуется послать через скрытый канал пару $(\Phi, c)$, а для передачи последующих ключей будет достаточно передавать лишь $c$. Впрочем, применение этого алгоритма в схеме из раздела 6 также является достаточно перспективным.

**Заключение**

Использование коммерческих криптографических средств, как аппаратных, так и программных, приводит к тому, что рядовой пользователь или даже фирма среднего размера не в состоянии удостовериться в «чистоте» используемых технологий. Программы, как правило, стараются максимально защитить от дизассемблирования, а анализ внутренней структуры аппаратных решений сам по себе весьма трудоемок и дорог, даже если производитель не применяет специальных защитных мер.

В работе было описано несколько методов, позволяющих производителю криптографического устройства или программы получать ключи пользователей. Все они вполне реальны, хоть и обладают несколько различными наборами свойств. Также следует отметить, что описанные методы применимы как к ECDSA, так и к большинству других схем ЭЦП, основанных на разновидностях задачи дискретного логарифмирования.

Возможным решением является использование программного обеспечения с открытыми исходными кодами — это один из немногих путей, позволяющих практически исключить наличие «закладок». К сожалению, в случае необходимости использования аппаратных средств подобного решения не существует.

## СПИСОК ЛИТЕРАТУРЫ:

1. Digital Signature Standard (DSS). FIPS 186-2. Federal Information Processing Standards Publication. U.S. Department of Commerce / National Institute of Standards and Technology. 27.01.2000.
2. *Young A., Yung M.* Malicious Cryptography Exposing cryptovirology. Wiley Publishing, Inc., 2004.
3. *Brassard* G. Modern Cryptology, a Tutorial, Lect. Notes Comput. Sci. Springer-Verlag, Berlin, 1988.
4. SEC 1: Elliptic Curve Cryptography // Certicom Research. 20.09.2000.