A Novel Related Nonce Attack for ECDSA

Marco Macchetti

Kudelski Security, Switzerland
marco.macchetti@kudelskisecurity.com

Abstract. We describe a new related nonce attack able to extract the original signing key from a small collection of ECDSA signatures generated with weak PRNGs. Under suitable conditions on the modulo order of the PRNG, we are able to attack linear, quadratic, cubic as well as arbitrary degree recurrence relations (with unknown coefficients) with few signatures and in negligible time. We also show that for any collection of randomly generated ECDSA nonces, there is one more nonce that can be added following the implicit recurrence relation, and that would allow retrieval of the private key; we exploit this fact to present a novel *rogue nonce attack* against ECDSA. Up to our knowledge, this is the first known attack exploiting generic and unknown high-degree algebraic relations between nonces that do not require assumptions on the value of single bits or bit sequences (e.g. prefixes and suffixes).

Keywords: ECDSA · PRNG · nonce attack.

1 Introduction

Digital signatures are nowadays ubiquitous and find application in secure communication protocols, code authentication, public key infrastructures and blockchain technologies. Control of funds deposited in any digital wallet is eventually based on the capability of its owner to prove ownership and sign transactions using an associated private key in a digital signature scheme. One of the most widely adopted digital signature standards, based on elliptic curve cryptography (ECC), is ECDSA [6]. Due to the high performance and small signature and key size, ECDSA has been adopted in many applications, from TLS to Bitcoin. A study [29] conducted in early 2021 on the top market-cap 100 blockchains showed that 74 coins use ECDSA as digital signature scheme (over the **secp256k1** curve [14]).

The security of ECC is based on the difficulty of solving the discrete logarithm problem over the group of points of a suitable elliptic curve (ECDLP) and its related decisional variants [28]; however in practice, the theoretical security level can be degraded by the possibility to collect side channel information during the process of cryptographic operations as it has been first demonstrated on symmetric key algorithms [16] and then extended to public key algorithms [7]. Indeed, physical leakage of secret data processed by embedded devices during sensitive operations, cache and memory accesses on mobiles [9], servers and desktops [26], combined with mathematical techniques [10][17] allows to get around

the hard ECDLP problem and to break security. In the case of ECDSA, this often translates to a compromise of the private key.

The critical operation in ECC is the point-scalar multiplication, i.e. the operation that benefits from DLP-hardness. This can be viewed as the equivalent of modular exponentiation in RSA schemes. The scalar k which multiplies the curve points can play different roles depending on the high level scheme which is instantiating the point multiplication operation, but almost often its confidentiality is a primary concern. In Diffie-Hellman protocols, it represents a party's private key (being ephemeral or static). In signature schemes such as ECDSA and Schnorr signatures [23], it is assumed to be a cryptographically strong random number (the *nonce*), and its confidentiality, integrity and non-repeatability must always be assured.

It is well known that if the value of even a single nonce used to generate an ECDSA signature is leaked, then anyone can retrieve the private key used to generate the signatures. Therefore, all nonces must be kept secret. It is also known that even if all nonces are kept secret, but a nonce value is reused for more than one signature, the private key can also be immediately retrieved. Such nonce collisions are very easy to detect, as the first half of each ECDSA signature depends only on the nonce value (it can in fact be viewed as the public ephemeral key, or a public commitment to the nonce value used to generate the second half of the signature). Researches have also shown how biased generation of nonces can lead to unveiling of the private key, provided a sufficient number of signatures is available (more details are given below).

1.1 Our Contributions

We advance the state of the art by proposing a novel attack that exploits high-degree relationships among the random values (nonces) used to generate several signatures, allowing to retrieve the signer's private key. Our method is not exploiting any side channel information coming from the signature generation process, but is rather a cryptanalytic attack that can be run, under certain assumptions, on a relatively small set of signatures generated by a given private key.

In our investigations, we make the assumption that nonces used to generate several ECDSA signatures are subject to an unknown polynomial relationship modulo n, where n is the order of the curve's generator point; more precisely, we suppose that they satisfy a recurrence polynomial equation of arbitrary degree and unknown coefficients modulo n. In this case it is very easy to recover the private key using only few signatures, without the need of lattice reduction.

The first step we need to perform is to find a way to get rid of the unknown coefficients involved in the recurrence equation; to this purpose, we devise a recursive algorithm that is able to rewrite the unknown recurrence relation as a polynomial involving only differences of nonces. Then, we can use the relationships between the nonces and the private key given by the second half of each signature to rewrite the polynomial only in terms of the private key. Finally, we

show that we can use known algorithms to find the roots of the polynomial; the private key will always be obtained as a root of the polynomial.

The attack works on all prime curves currently standardized and in use for ECDSA (including Bitcoin curve **secp256k1**) and up to our knowledge, it is the first to exploit relationships of degrees higher than linear.

Cases where nonces are generated with Linear Congruential Generators (LCG), quadratic or cubic generators with unknown coefficients modulo n, are special cases of this general category of relationships. After showing how to attack nonces generated with an arbitrary recurrence relation, we then prove that even sets of fully random nonces allow the attacker to retrieve the private key when the set is completed by one additional nonce that follows the (unknown) recurrence relation given by the random nonces taken in a given order (we refer to this as the *rogue nonce* example). When a sufficient number of signatures are generated with a given private key, there will always be a reordering of the signatures that make the recurrence attack work, leading to compromise of the private key. However, we are not able to find this ordering faster than brute-force.

1.2 Related Work

The generation of an ECDSA signature is arguably a fragile process, requiring the creation and use of cryptographically strong random values; it is understandable that its security has been widely scrutinized and that attacks exploiting bad implementations have been extensively published [22], [17], [24], [8].

In the ANSI X9.62 document [31], the group where the DLP-hard operation takes place is the group on the curve defined by the generator point G, which has order n. Therefore it seems then natural to assign the requirement that $k \in [1, n-1]$, where n is the order of the generator point of the elliptic curve. This constraint is coherent with the fact that private keys in elliptic curve key pairs must be chosen uniformly randomly over the same interval (in the document the per-message secret k is indeed referred-to as an ephemeral key).

When k is chosen, a scalar-point multiplication is performed to obtain the corresponding public key $R = [k]G$ and the signature r part is determined as the x-coordinate of R. To mitigate the usefulness of partial information obtained from the process of computing $R = [k]G$, the value of k can be blinded by preliminary addition of a random multiple of n, where the multiplication factor must have at least $\frac{\log_2 n}{2}$ bits [33]. The partial information could be an MSB prefix of the scalar k, that can be used to retrieve the private key by solving instances of the hidden number problem [12] by means of lattice reduction [27].

Even if the ECDSA signature generation is performed securely, if the nonce values k are not properly generated in a random way, attacks become possible. An example would be if the k values are biased, that is if their distribution over the interval $[1, n-1]$ is not uniform. for example if they contain known prefixes, suffixes or common bit sequences [18], or if they are generated using non cryptographic PRNGs [13],[21],[8]. This latter case is particularly interesting; if nonces are generated using Linear Congruential generators (LCGs), the signatures can

be likely used to mount an attack by means of Lattice Reduction algorithms [19], and the signer's key can be obtained.

The biases used in [19], [13] and exploited so far in the literature can be seen as simple relations between the nonce bits, or linear relations between the full nonce values. Up to our knowledge, there are no published studies on cases where the nonce values are related in a more complicated way, for instance by means of quadratic, cubic or higher-degree algebraic relationships. This will be our goal, and is motivated by the fact that examples of non-cryptographic PRNGs include quadratic and cubic congruential generators [32].

The rest of this paper is organized as follows: Section 2 covers preliminaries useful to understand the rest of the paper; Section 3 presents the attack; Section 4 discusses an implementation and give some results, Section 5 concludes the paper.

2 Preliminaries

In this section we give the required preliminaries that are going to be used in the rest of this work. In the following, we use "iff" as "if and only if", DLP as "discrete logarithm problem" and ECDLP as "elliptic curve discrete logarithm problem". By *algorithm* or *procedure* we mean a uniform family of circuits of depth and width polynomial in the index of the family. Namely: $\mathcal{F} = \{\mathbf{F}_n\}_{n\in\mathbb{N}}$ is an algorithm iff there exists an integer $\bar{n}$, polynomials τ, w, d and a Turing machine $\mathbf{M}_{\mathcal{F}}$ which, given n as input, outputs a description of $\mathbf{F}_n$ in time at most $\tau(n)$ for any $n > \bar{n}$, where $\mathbf{F}_n$ is a Boolean circuit of width at most $w(n)$ and depth at most $d(n)$. If an algorithm A is deterministic, we denote its output y on input x as $y := A(x)$, while if it is randomized we use $y \leftarrow A(x)$. We will also use $x \xleftarrow{\mathcal{D}} X$ to denote that an element x is sampled from a distribution $\mathcal{D}$ over a set X (or simply $x \leftarrow \mathcal{D}$ when the image set is implied); or we will write $x \xleftarrow{\$} X$ if x is sampled uniformly at random from a set X. We will call *negligible* (and denote by $\mathsf{negl}(n)$) a function that grows more slowly than any inverse polynomial in n, and *overwhelming* a function which is 1 minus a negligible function. Finally, $a\|b$ denotes concatenation of strings.

2.1 Cryptographic Primitives

A Weierstrass elliptic curve E defined over a finite field $\mathbb{F}_q$ is a set of points $P = (x, y)$ where x and y belongs to $\mathbb{F}_q$ and satisfy a certain equation curve E, together with the *point at infinity* denoted by O :

- If $q = p$, $p > 3$ a prime number, every point $P = (x, y)$, $P \neq O$, must satisfy the following equation in $\mathbb{F}_p$:

$$E : y^2 \;=\; x^3 + ax + b \tag{1}$$

  where the parameters $a, b \in \mathbb{F}_p$ are such that the discriminant $\Delta = 4a^3 + 27b^2$ is not equal to zero in $\mathbb{F}_p$.

– If $q = 2^m$ with m a prime number, every point $P = (x, y)$, $P \neq O$, must satisfy the following equation in $\mathbb{F}_{2^m}$:

$$E : y^2 + xy \ = \ x^3 + ax^2 + b \tag{2}$$

where the parameter $b \neq 0$ in $\mathbb{F}_{2^m}$.

The group law of elliptic curves is defined by the addition of two points on the curve (shown for prime curves, see [20]):

$$x_3 = \frac{y_2 - y_1}{x_2 - x_1}^2 - x_1 - x_2 \ \text{ and } \ y_3 = \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x_3) - y_1 \tag{3}$$

Doubling of a point is similarly obtained as:

$$x_3 = \frac{3x_1^2 + a}{2y_1}^2 - 2x_1 \ \text{ and } \ y_3 = \frac{3x_1^2 + a}{2y_1}(x_1 - x_3) - y_1 \tag{4}$$

To construct cryptographic schemes using elliptic curves, domain parameters $D = \{q, a, b, G, n, h\}$ must be defined consisting of following elements:

– A field order q
– An elliptic curve defined over $\mathbb{F}_q$ by two coefficients $a, b \in \mathbb{F}_q \times \mathbb{F}_q$
– A *base point* G, defined by its affine coordinates $(x_G, y_G) \in \mathbb{F}_q \times \mathbb{F}_q$ which is a point of $E(\mathbb{F}_q)$. G has prime order.
– The order n of G
– The *co-factor* $h = \frac{\#E(\mathbb{F}_q)}{n}$

The operation called *scalar point multiplication* is constructed upon the elliptic curve group operation of point addition:

$$R = [k]P \tag{5}$$

where k is an integer and P a point of $E(\mathbb{F}_q)$.

This operation is directly linked to the ECDLP and provides security in all cryptographic schemes based on ECC. Its execution time and its sensitivity regarding physical attacks are major concerns when ECC is implemented in HW and/or SW. Several regular algorithms are available to execute securely the scalar point multiplication, for details we refer to [15] and [20].

2.2 ECDSA

Let us represent the ECDSA domain parameters as $D_{ECDSA} = (q, a, b, G, n, h)$, the key pair be (d, Q) with $Q = [d]G$ and the required hash function being $h(.)$. The process of generating an ECDSA signature is given in Algorithm 1.

To produce digital signatures, a random or pseudo-random per-signature secret k must be generated, consumed and discarded. It can also be deterministically generated [30] , but in any case its secrecy must be guaranteed because its leakage enables to retrieve the signer's private key.

Algorithm 1 ECDSA signature generation

Require: D_{ECDSA}, private key d, message m
returns signature := (r, s)
Step 1: $k \xleftarrow{\$} [1, n-1]$
Step 2: $R = [k]G = (x_R, y_R)$
Step 3: $r = x_R \bmod n$, if $r = 0$ go to Step 1.
Step 4: $e = h(m)$
Step 5: $s = k^{-1}(e + d \times r) \bmod n$, if $s = 0$ go to Step 1
Return (r, s)

One quickly realizes that the sensitivity of k is equivalent to the sensitivity of the signer's private key, and therefore special care must be taken versus implementation attacks, including physical attacks, when k is manipulated.

During signature generation, k is used three times. The first time is to compute r via scalar point multiplication, the second time to obtain its modular inverse $k^{-1} \bmod n$ and the last time in its inverse form during s computation. None of these operations should leak any information on the bits of k.

3 The New Attack

3.1 A Generic Observation

Suppose that N signatures are generated using the ECDSA algorithm using the same private key d; let us denote the parameters of the i-th signature as (k_i, h_i, r_i, s_i) where k denotes the nonce, h the message hash, r the first half of the signature and s the second half of the signature. All equations in the rest of this paper are intended modulo n, where n is the order of the curve's generator point G. For each generated signature we can write:

$$k_i = \frac{h_i}{s_i} + \frac{r_i}{s_i} d \tag{6}$$

In other words, the second half of every signature gives us a linear relation between the used nonce and the private key, as the message hash value and the two halves of each signature are public and known. We believe this is, by itself, an interesting observation. As the first half of every signature is the x coordinate of the point $R_i = [k_i]G$, it is straightforward to use the curve equation to obtain the point R_i (up to a sign of its y coordinate, here we can make the assumption of taking the smallest value of y). Obtaining k_i from R_i would mean solving the discrete logarithm problem over the curve; but indeed using the second halves of the signatures, we can write a relationship between the nonce values. Taking the example case of two signatures, it is easy to verify that we can re-write (6) as:

$$k_1 = \frac{r_1 s_0}{r_0 s_1} k_0 + \frac{h_1 r_0 - h_0 r_1}{r_0 s_1} \tag{7}$$

and, multiplying times the generator point G, we obtain (regrouping known values in a, b):

$$R_1 = [a]R_0 + [b]G \tag{8}$$

which is a non-trivial relationship that one is not supposed to write knowing only the nonce commitments (i.e. the points R_i).

This shows that given the set of N points R_i, we are not confronted with N instances of the discrete logarithm problem, but with only *one* instance, as we know how all these nonce values are related, through the private key d by means of the values of s_i. This is, after all, the reason behind the well known fact that knowledge of one ECDSA nonce value is equivalent to knowledge of the private key, and also to knowledge of all nonce values ever used to generate signatures with that particular private key.

When the nonces used for N signatures obey a multivariate polynomial equation, we can rewrite the equation as a univariate polynomial involving only the private key and public values by substituting the relations above. For example, if we know coefficients a_i and exponents e_i such that

$$a_0 k_0^{e_0} + a_1 k_1^{e_1} + a_2 k_2^{e_2} + \cdots + a_N = 0 \tag{9}$$

then we can re-write it as:

$$a_0(\frac{h_0}{s_0} + \frac{r_0}{s_0}d)^{e_0} + a_1(\frac{h_1}{s_1} + \frac{r_1}{s_1}d)^{e_1} + a_2(\frac{h_2}{s_2} + \frac{r_2}{s_2}d)^{e_2} + \cdots + a_N = 0 \tag{10}$$

Now, this is a polynomial of degree $\max_i(e_i)$ in the unknown d. Algorithms exist to quickly expand and find roots of such polynomials over finite fields (for instance, Berlekamp's algorithm[11]) and are often implemented in freely available computer algebra packages, such as SageMath [5] for instance; the private key d will always appear among the roots of the polynomial. Note that more complex relations where the nonces are multiplied together will also work, as for example

$$a_0 k_0^{e_0 0} k_1^{e_0 1} \cdots k_{N-1}^{e_{0N-1}} + a_1 k_0^{e_1 0} k_1^{e_1 1} \cdots k_{N-1}^{e_{1N-1}} + \cdots + a_N = 0 \tag{11}$$

can also be re-written as above and will also lead to a univariate polynomial in d (with potentially higher degree). The natural question that arises is how to find such relation between the nonces; if they are chosen in a properly random way, this is generally not possible. However, in some cases it may be possible to write such relation, as we will see in the next sections.

3.2 Nonces generated with an (unknown) recurrence equation

Let us consider the case where the signer generates N ECDSA signatures choosing the first nonce k_0 at random and all the following $N-1$ nonces using a $(N-3)$-degree recurrence relation with $N-2$ unknown coefficients. We can think about this as the most general case of a polynomial random number generator with secret coefficients a_i, seeded by the initial state k_0.

That is, we can write:

$$\begin{aligned}
k_1 &= a_{N-3}k_0^{N-3} + a_{N-4}k_0^{N-4} + \cdots + a_1k_0 + a_0 \\
k_2 &= a_{N-3}k_1^{N-3} + a_{N-4}k_1^{N-4} + \cdots + a_1k_1 + a_0 \\
k_3 &= a_{N-3}k_2^{N-3} + a_{N-4}k_2^{N-4} + \cdots + a_1k_2 + a_0 \\
&\cdots \\
k_{N-1} &= a_{N-3}k_{N-2}^{N-3} + a_{N-4}k_{N-2}^{N-4} + \cdots + a_1k_{N-2} + a_0
\end{aligned} \tag{12}$$

What we want is to write a polynomial involving only the nonces, and to get rid of the unknown recurrence relation coefficients a_i , so that we can then rewrite the equation with only the private key. Let us tackle the general case by analyzing the simplest cases first, and then proceed with a recursive algorithm. When $N = 4$, we are essentially looking at the simple case of an LCG with unknown multiplier a_1, increment a_0 and initial state k_0, that is:

$$\begin{aligned}
k_1 &= a_1k_0 + a_0 \\
k_2 &= a_1k_1 + a_0 \\
k_3 &= a_1k_2 + a_0
\end{aligned} \tag{13}$$

To get rid of coefficient a_0 we subtract the second equation from the first, and the third from the second:

$$\begin{aligned}
k_1 - k_2 &= a_1(k_0 - k_1) \\
k_2 - k_3 &= a_1(k_1 - k_2)
\end{aligned} \tag{14}$$

Now, let us denote the difference of two nonces $k_i - k_j$ as $k_{i,j}$:

$$\begin{aligned}
k_{1,2} &= a_1k_{0,1} \\
k_{2,3} &= a_1k_{1,2}
\end{aligned} \tag{15}$$

or, leaving a_1 on both right sides:

$$\begin{aligned}
\frac{k_{1,2}}{k_{0,1}} &= a_1 \\
\frac{k_{2,3}}{k_{1,2}} &= a_1
\end{aligned} \tag{16}$$

and again subtracting the second from the first equation finally gives:

$$\frac{k_{1,2}}{k_{0,1}} - \frac{k_{2,3}}{k_{1,2}} = 0 \tag{17}$$

or

$$k_{1,2}^2 - k_{2,3}k_{0,1} = 0 \tag{18}$$

This can be easily re-written as a second degree polynomial in d with known coefficients by substituting all occurrences of $k_{i,j}$ with:

$$k_{i,j} = (\frac{r_i}{s_i} - \frac{r_j}{s_j})d + \frac{h_i}{s_i} - \frac{h_j}{s_j} \tag{19}$$

The private key can then be easily recovered as one of the roots of this polynomial.

This means that despite the fact that the signer is using a Linear Congruential Generator (LCG) modulo n with secret coefficients and initial state, we are able to retrieve his private key by only observing four signatures, in negligible time. Lattice reduction algorithms are also able to obtain the private key in the linear case so let us now take a look at higher degrees recurrences.

For $N = 5$, the recurrence equations are:

$$\begin{aligned} k_1 &= a_2k_0^2 + a_1k_0 + a_0 \\ k_2 &= a_2k_1^2 + a_1k_1 + a_0 \\ k_3 &= a_2k_2^2 + a_1k_2 + a_0 \\ k_4 &= a_2k_3^2 + a_1k_3 + a_0 \end{aligned} \tag{20}$$

and what we obtain in the end after a similar elimination of coefficients is:

$$(k_{1,2}^2 - k_{2,3}k_{0,1})k_{1,3}k_{2,3} - (k_{2,3}^2 - k_{3,4}k_{1,2})k_{0,1}k_{0,2} = 0 \tag{21}$$

which leads to a 4-degree polynomial in d after the usual substitution; again, d is found as a root of the polynomial. This is representing the case of a Quadratic Congruential Generator (QCG) modulo n with secret coefficients and unknown initial state.

Similarly, for a Cubic Congruential Generator (CCG) we set $N = 6$ and we obtain a 7-degree equation:

$$\begin{aligned} &((k_{1,2}^2 - k_{2,3}k_{0,1})k_{1,3}k_{2,3} - (k_{2,3}^2 - k_{3,4}k_{1,2})k_{0,1}k_{0,2})k_{1,4}k_{2,4}k_{3,4} \\ -&((k_{2,3}^2 - k_{3,4}k_{1,2})k_{2,4}k_{3,4} - (k_{3,4}^2 - k_{4,5}k_{2,3})k_{1,2}k_{1,3})k_{0,1}k_{0,2}k_{0,3} = 0 \end{aligned} \tag{22}$$

Indeed, we can show that the polynomials involving the nonces generated by (12) can be obtained using a recursive algorithm that constructs polynomials using the nonces differences $k_{i,j}$; then, substituting the values with (19) leads to a polynomial in d that can be solved to obtain the private key.

The recursive algorithm has been constructed by inspection and generalization of the hand-constructed polynomials shown above for $N = 4, 5, 6$ and has been tested to work for N up to 16. The degree in d of the polynomial for N signatures is equal to $1 + \sum_{i=1}^{N-3} i$. The polynomial in terms of the $k_{i,j}$ for N signatures can be obtained by calling the recursive function $dpoly(N-4, N-4, 0)$ where the function *dpoly* is described in Algorithm 2.

Algorithm 2 dpoly(n, i, j)

```
if i == 0 then
    return k^2_{j+1,j+2} − k_{j+2,j+3} k_{j,j+1}
else
    left = dpoly(n, i − 1, j)
    for m = 1 to i + 2 do
        left = left·k_{j+m,j+i+2}
    end for
    right = dpoly(n, i − 1, j + 1)
    for m = 1 to i + 2 do
        right = right·k_{j,j+m}
    end for
    return (left - right)
end if
```

3.3 Adding a carefully crafted signature to an existing set

Let us now consider the case where the signer has correctly generated $N-1$ nonces in a random way, and has so produced a set of $N-1$ signatures. The considerations of the previous section are not valid for this set, but if we take a look at (12) we can see that they describe a rather generic recurrence equation. The fact that the coefficients are unknown and we do not make assumptions on them, lead to the following consideration: let us take all equations from (12) but the last one. We have a set of $N-2$ equations binding the $N-1$ nonces via the $N-2$ coefficients a_i. Therefore, for a given set of such $N-1$ randomly generated nonces, one can always obtain one solution, i.e. one set of coefficients a_i that binds them with a recurrence relation of degree $N-3$.

In other words, even for a set of random nonces, there always exists an implicit unknown recurrence equation of the form (12) that binds them. Now, if we complete the set of signature with an additional one, where the nonce is chosen to satisfy the recurrence equation, we are back in the case analyzed in the previous section, and we can simply obtain the signer's private key using the technique explained above.

We can refer to this additional nonce as the *rogue nonce*. Note that by changing the order of the first $N-1$ signatures, one obtains different values for the implicit recurrence relation coefficients, and therefore a different value of the rogue nonce. When the value of N has the order of magnitude of the number of bits of the curve, it is easy to see that given a set of properly generated N signatures, it is always possible to re-order the first $N-1$ ones to make the last signature fit the underlying implicit recurrence relation.This follows from the fact that the number of permutations grows with the factorial or approximately N^N, which is greater than 2^N. Therefore given a big enough set of signatures, there will always be a given re-ordering that allows the retrieval of the private key by means of the attack proposed above. All of this is only of theoretical interest, as we are not able here to devise a way of finding this re-ordering easier than with brute-force.

More formally, given a set of $N-1$ nonces, the coefficients a_i that relate them can be obtained by writing the recurrence relations (12) from 1 to $N-2$ in matrix form, and by inverting the $(N-2)\times(N-2)$ matrix to obtain:

$$\begin{bmatrix} a_{N-3} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} k_0^{N-3} & \dots & k_0^0 \\ \vdots & \ddots & \vdots \\ k_{N-3}^{N-3} & \dots & k_{N-3}^0 \end{bmatrix}^{-1} \begin{bmatrix} k_1 \\ \vdots \\ k_{N-2} \end{bmatrix} \tag{23}$$

As we can see,knowledge of the nonces is necessary to derive them, meaning the value of the rogue nonce is uniquely determined by the set of the previous $N-1$ nonces and by their ordering.

4 Implementation And Benchmarks

The techniques presented in this paper have been implemented in Python language, using the SageMath Python extensions. SageMath [5] is a powerful tool that is able to execute Python programs in a native way and gives to the programmer a powerful set of algebraic manipulation routines; these can be used to construct polynomials, manipulate them and find roots.

The code for the related nonces and the rogue nonce cases can be found in this GitHub repository [1], under the `original_attack` folder. To execute them, it is necessary to invoke Sage and to install the Python ecdsa, hashlib and random libs. Retrieval of the private key from a set of signatures takes roughly 70 ms on average a common laptop for the sec256k1 curve, about 340 ms on average for the NIST secp384r1 curve, and about 490 ms on average for the secp521r1 curve; these numbers do not vary significantly for linear, quadratic or cubic recurrence equations. When $N = 16$, the algorithm takes about 6.5 s to retrieve the private key.

We used this code as a basis to conduct a scan of the Bitcoin and Ethereum blockchains, to spot possible usages of such weak PRNGs for ECDSA signature generation. We were not able to obtain results, but we re-discovered cases of usage of repeated nonces. Indeed, a repeated nonce is a trivial case of the recurrence relation studied in this paper, where the only non-zero coefficient is a_0, which is equal to the repeated nonce; details on our experiments can be found here.

Complete source code can be found in the GitHub repositories [1], [2] for Bitcoin ECDSA signatures dumping code, [3] for Ethereum ECDSA signatures dumping code. We also tried the attack without results on dumps of TLS connection establishments, see the GitHub repository [4] for more details.

5 Conclusions and Future Directions

In this paper we presented an attack that exploits the existence of generic nonlinear (high-degree) algebraic relations between ECDSA nonces in order to retrieve

the signer's private key. We studied the case of a generic recurrence relation of degree $N-3$ with unknown coefficients, showing that N such signatures allow the attacker to obtain the private key in negligible time. We have also shown that given a sufficiently large collection of signatures generated with uniformly random nonces, there is an ordering of the signatures which would allow easy retrieval of the private key by means of the attack proposed here.

We see several lines of extension of our work. First, we would like to remark that the considerations presented here certainly apply also to other signature schemes that require the generation of a random nonce (or ephemeral secret), for instance Schnorr signatures. Deterministic variants (e.g. deterministic ECDSA and EdDSA [25]) make use of cryptographic hash functions to generate the nonces and are thus inherently resistant to the attacks described here.

It would be interesting to devise techniques that would allow retrieval of the private key in the case where the modulo used by the recurrence relation is different from the order of the curve, as this seems an interesting case for known LCG, QCG and CCG such as the ones referenced in [32]. Even if it would probably be necessary to make further assumptions, for instance that the moduli should be co-prime, we consider this as the most interesting evolution of the work presented here.

It would be nice to devise a way to find signature reordering faster than with brute-force, for the case where a rogue nonce is considered. For experimental trials, we were rather limited by the latency of the process, especially because the proposed attack works only if the generated signatures are taken in the same order with which they have been chronologically generated (or the exact opposite one); it would be interesting to try to extend the search scope, possibly targeting other blockchains and/or protocols. It could also be possible to avoid the use of SageMath, and to code the attack natively with low-level languages, allowing compilation and optimization; this would probably boost the speed for a widespread search for weaknesses.

A negative check for the success of the attack could be integrated for instance in existing libraries [19]; the goal would be to exclude such conditions that would allow compromising the signer's private key.

5.1 Acknowledgements

The author would like to thank Nils Amiet for his work on the implementation of the attack and the tests against the blockchain signatures (see Section 4) and Tommaso Gagliardoni and Karine Villegas for the fruitful discussions and for revising preliminary versions of this paper.

References

1. https://github.com/kudelskisecurity/ecdsa-polynomial-nonce-recurrence-attack
2. https://github.com/kudelskisecurity/ecdsa-dump-bitcoin
3. https://github.com/kudelskisecurity/ecdsa-dump-ethereum

4. https://github.com/kudelskisecurity/ecdsa-dump-tls
5. Sagemath project homepage, https://www.sagemath.org/
6. FIPS186-5, Digital Signature Standard (DSS). National Institute of Standards and Technologies edn. (February 2023), https://csrc.nist.gov/publications/detail/fips/186/5/final
7. Amiel, F., Feix, B., Villegas, K.: Power analysis for secret recovering and reverse engineering of public key algorithms. In: Adams, C.M., Miri, A., Wiener, M.J. (eds.) Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers. Lecture Notes in Computer Science, vol. 4876, pp. 110–125. Springer (2007)
8. Aranha, D.F., Novaes, F.R., Takahashi, A., Tibouchi, M., Yarom, Y.: Ladderleak: Breaking ecdsa with less than one bit of nonce leakage. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. p. 225–242. CCS '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3372297.3417268, https://doi.org/10.1145/3372297.3417268
9. Belgarric, P., Fouque, P., Macario-Rat, G., Tibouchi, M.: Side-channel analysis of weierstrass and koblitz curve ECDSA on android smartphones. In: Sako, K. (ed.) Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9610, pp. 236–252. Springer (2016)
10. Benger, N., van de Pol, J., Smart, N.P., Yarom, Y.: "ooh aah... just a little bit" : A small amount of side channel can go a long way. In: Batina, L., Robshaw, M. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8731, pp. 75–92. Springer (2014)
11. Berlekamp, E.R.: Factoring polynomials over large finite fields. Mathematics of Computation **24**, 713–735 (1970)
12. Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in diffie-hellman and related schemes. In: Koblitz, N. (ed.) Advances in Cryptology — CRYPTO '96. pp. 129–142. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)
13. Breitner, J., Heninger, N.: Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies. In: Goldberg, I., Moore, T. (eds.) Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11598, pp. 3–20. Springer (2019). https://doi.org/10.1007/978-3-030-32101-7_1, https://doi.org/10.1007/978-3-030-32101-7_1
14. Brown, D.: Sec 2: Recommended elliptic curve domain parameters (2010), http://www.secg.org/sec2-v2.pdf
15. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity. IEEE Transactions on Computers **53**(6), 760–768 (2004)
16. C.Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M.J. (ed.) Advances in Cryptology, 19th Annual International Cryptology Conference – CRYPTO '99. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999)

17. De Mulder, E., Hutter, M., Marson, M.E., Pearson, P.: Using bleichenbacher"s solution to the hidden number problem to attack nonce leaks in 384-bit ecdsa. In: Bertoni, G., Coron, J.S. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2013. pp. 435–452. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
18. Faugère, J.C., Goyet, C., Renault, G.: Attacking (ec)dsa given only an implicit hint. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography. pp. 252–274. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
19. Google: Project paranoid – checks for weaknesses in ecdsa generation, https://github.com/google/paranoid_crypto/blob/main/docs/ecdsa_signature_tests.md
20. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer New York, NY (2004)
21. Heninger, N.: Rsa, dh, and DSA in the wild. IACR Cryptol. ePrint Arch. p. 48 (2022), https://eprint.iacr.org/2022/048
22. Howgrave-Graham, N.A., Smart, N.P.: Lattice Attacks on Digital Signature Schemes. In: Designs, Codes and Cryptography volume 23. pp. 283–290 (2001)
23. fur Sicherheit in der Informationstechnik, B.: Elliptic curve cryptography - technical guideline tr-03111-v2.0 (2012)
24. Jancar, J., Sedlacek, V., Svenda, P., Sys, M.: Minerva: The curse of ECDSA nonces (systematic analysis of lattice attacks on noisy leakage of bit-length of ECDSA nonces). IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(4), 281–308 (2020). https://doi.org/10.13154/tches.v2020.i4.281-308
25. Josefsson, S., Liusvaara, I.: Edwards-Curve Digital Signature Algorithm (EdDSA), RFC8032-IETF edn. (January 2017)
26. Kaufmann, T., Pelletier, H., Vaudenay, S., Villegas, K.: When Constant-time Source Yields Variable-Time Binary: Exploiting Curve25519-Donna Built with MSVC 2015. In: Sara Foresti, G.P. (ed.) Cryptology and Network Security. Lecture Notes in Computer Science, vol. 10052, pp. 573–582. Springer (2016)
27. Lenstra, A., Lenstra, H., Lovasz, L.: Factoring polynomials with rational coefficients. MATH. ANN **261**, 515–534 (1982)
28. Menezes, A.: The elliptic curve discrete logarithm problem: State of the art. In: Matsuura, K., Fujisaki, E. (eds.) Advances in Information and Computer Security. pp. 218–218. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
29. Nash: Cryptography behind the top 100 cryptocurrencies, http://ethanfast.com/top-crypto.html
30. Pornin, T.: Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA), RFC6979-IETF edn. (August 2013)
31. Public Key Cryptography For The Financial Services Industry: X9-62-2005 The Elliptic Curve Digital Signature ECDSA, American National Standards Institute edn. (November 2005)
32. Saarinen, M.O.: Sp 800–22 and gm/t 0005–2012 tests: Clearly obsolete, possibly harmful. In: 2022 IEEE European Symposium on Security and Privacy Workshops. pp. 31–37. IEEE Computer Society, Los Alamitos, CA, USA (jun 2022). https://doi.org/10.1109/EuroSPW55150.2022.00011, https://doi.ieeecomputersociety.org/10.1109/EuroSPW55150.2022.00011
33. Schindler, W., Wiemers, A.: Efficient Side Channel Attacks on Scalar Blinding on Elliptic curves with Special Structure. ECC workshop (2015)