Attacks on ECDSA when the nonces are generated with a weak pseudo random number generator

Pedro Miranda
pedro.c.miranda@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

July 2019

Abstract

In this article we describe various attacks against the ECDSA based on the lattice attack used to solve the hidden number problem. We start by presenting the usual lattice attack against the ECDSA, then we design two new attacks against the ECDSA, the first one recovers a linear equation between two different private keys and the second one uses a solution to the two variables hidden number to recover two private keys. Moreover, we describe how to use that two variables lattice attack to design an attack which recovers a single private key, which, according to our results, has a better probability of success than the usual one variable lattice attack. These attacks were implemented and tested for the secp256k1 curve, the curve used in Bitcoin's public key-cryptography, using the LLL algorithm.
Keywords: ECDSA, Hidden Number Problem, Lattice, LLL, BKZ.

1. Introduction

Due to the increase in the amount of online transactions, digital signatures are being used more frequently than ever. One of the most popular schemes is the Elliptic Curve Digital Signature Algorithm (ECDSA) [1] which is used, for example, in Bitcoin transactions [13]. In order to ensure that the ECDSA is secure against total break attacks, many attacks have been designed that try to recover a user's private key. Some of those attacks start by mapping the problem of finding the user's private key to a hidden number problem (HNP) [6] and then, using a lattice reduction algorithm such as LLL [11] or BKZ [15]. These are polynomial time attacks and they can recover the private key if there is a side-channel attack which leaks a portion of the nonces' bits or if there is a known weakness in the pseudorandom number generator (PRNG) used to generate the nonces.

In this article we will study lattice attacks which will use the solution to HNP in [7] to recover a private key, a linear equation between two different private keys or two private keys from a list of ECDSA signatures. The attacks which recover a linear equation between two different private keys and the one which uses a solution to the two variables hidden number to recover two private keys are new attacks. Moreover, we describe how to use that two variables lattice attack to design an attack which recovers a single private key, which, according to our results, has a better probability of success than the usual one variable lattice attack.

In section 2 we will give some background in elliptic curves, ECDSA, lattices, the HNP for one and two variables and their lattice solution. In section 3 and 4 we present the attacks against the ECDSA. In section 5 we implement and test the lattice attacks. Lastly, in section 6 we analyze our results and discuss ideas for future works.

1.1. Related work

In [6] Boneh and Venkatesan solved the HNP by mapping it to a closest vector problem (CVP) and then using the LLL and Babai's nearest plane algorithm [2]. Moreover, they proved that if there was an oracle which leaked the most significant bits (MSB) of the secret key in a Diffie-Hellman key-exchange protocol [9], the problem of recovering the secret key can be reduced to the HNP.

In [10] Howgrave-Graham and Smart applied similar techniques to the ones in [6] to attack a 160-bit DSA assuming that there existed an oracle which recovered a portion of the nonces' bits.

In [14] Nguyen and Shparlinski presented the first polynomial-time algorithm that provably recovers the signer's secret ECDSA key when a few bits of the random nonces are known for a number of signatures.

In [3] the authors applied the FLUSH+RELOAD side-channel attack based on cache hits/misses to extract a small amount of data from OpenSSL ECDSA signature requests. Then, they applied a lattice attack to extract the private key. This side-channel attack was able to recover information from almost all of the observed nonces.

In [12] the authors described an attack against nonce leaks in 384-bit ECDSA using an FFT-based attack due to Bleichenbacher [4, 5].

In [7] the authors designed a lattice attack which was able to compute hundreds of Bitcoin private keys and dozens of Ethereum, Ripple, SSH, and HTTPS private keys.

2. Background

Let q, k be two integers. We define $q \bmod k$ as the only integer x such that $x \equiv q \bmod k$ and $0 \leq x \leq q-1$. Moreover, we define as $|q|_k$ as $\min\{q \bmod k, k - q \bmod k\}$.

2.1. Elliptic Curves

An elliptic curve E defined over a field $\mathbb{F}$, whose characteristic is neither 2 nor 3, is a smooth curve with equation

$$Y^2 = X^3 + aX + b^3, \quad a, b \in \mathbb{F}. \tag{1}$$

We define $E(\mathbb{F})$ as the set of points $(x, y) \in \mathbb{F}^2$ which are a solution to (1), along with a point at infinity O. We can define an operation $+$ in $E(\mathbb{F})$ such that $(E(\mathbb{F}), +)$ is an abelian group with identity O. Since $E(\mathbb{F})$ is an abelian group, given a point $P \in E(\mathbb{F})$ and an integer k, we can define the point kP.

Using the double and add method, kP can be computed using at most $2\lfloor \log k \rfloor$ elliptic curve additions. Let $p \geq 3$ be a prime number, then $\mathbb{Z}_p$ is a field and it can be proven that each elliptic curve addition in $\mathbb{Z}_p$ can be computed in time $O \log(p)^3$. Therefore,if E is an elliptic curve over $\mathbb{Z}_p$ and $k \leq p$, the operation kP can be computed in $O(\log(p)^4)$ time for any $P \in E(\mathbb{Z}_p)$.

2.2. ECDSA

Let E be an elliptic over $\mathbb{Z}_p$, where $p \geq 3$ is a prime number. Given P and Q in $E(\mathbb{Z}_p)$, if there exists an integer k such that $kP = Q$, k is called the discrete logarithm of Q base P.

Definition 2.1. Let P be a generator of $E(\mathbb{Z}_p)$. The discrete logarithm problem is the problem of given Q, finding k such that $kP = Q$.

There are no efficient algorithms known for the discrete logarithm problem. This fact can be used to design a signature scheme, the ECDSA. Let $p > 3$ be a prime number, E be an elliptic curve over $\mathbb{Z}_p$ and P a generator of $E(\mathbb{Z}_p)$ with order q, where q is a prime number. These are system-wide parameters. Each user Alice selects a random integer x in the interval $1 < x < q-1$ and computes $Q = xP$. Alice's private key is x and her public key is Q.

ECDSA Signature Generation. To sign a message m Alice does the following

1. She computes a random integer k in the interval $1 < k < q-1$.
2. She computes $kP = (x_1, y_1)$ and defines $r = x_1 \bmod q$. If $r = 0$, return to 1.
3. She computes $s = k^{-1}(H(m) + xr) \bmod q$, where $H(m)$ is the hash value of the message m. If $s = 0$, return to 1.
4. The signature is the pair (r, s).

Lemma 2.1. *Let E be an elliptic curve over a field $\mathbb{F}$ and P be a point in the elliptic curve with order q. If $kP = (x_1, y_1)$, then $(q-k)P = (x_1, -y_1)$.*

Proof. Given a point $(x, y) \in E(\mathbb{F})$ we have $-(x, y) = (x, -y)$, thus, for $kP = (x_1, y_1)$ we have $-kP = (x_1, -y_1)$, hence we need to prove that $(q-k)P = -kP$. Since the order of P is q, we have

$$\begin{aligned} & qP = O \\ \iff & kP + (q-k)P = O \\ \iff & (q-k)P = -kP. \end{aligned}$$

□

Theorem 2.2. *Let E be an elliptic curve over $\mathbb{Z}_p$, where p is a prime number such that $p \equiv 3 \bmod 4$. Then, for each $P = (x_1, y_1) \in E(\mathbb{Z}_p)$, we either have $y_1 \equiv \left(x_1^3 + ax_1 + b\right)^{\frac{p+1}{4}} \bmod p$ or $y_1 \equiv -\left(x_1^3 + ax_1 + b\right)^{\frac{p+1}{4}} \bmod p$.*

Proof. Since P is a point of the elliptic curve E, we have $y_1^2 \equiv x_1^3 + ax_1 + b \bmod p$. Moreover, for each x_1 there at most two solutions to $y^2 \equiv x_1^3 + ax_1 + b \bmod p$ since $\mathbb{Z}_p$ is a field. Thus, if we can prove that for the two values of y_1 presented in the theorem we have $y_1^2 \equiv x_1^3 + ax_1 + b \bmod p$ the proof is complete. For each of these values of y_1 we have

$$\begin{aligned} y_1^2 &\equiv \left(\left(x_1^3 + ax_1 + b\right)^{\frac{p+1}{4}}\right)^2 \mod p \\ &\equiv \left(x_1^3 + ax_1 + b\right)^{\frac{p+1}{2}} \mod p \\ &\equiv \left(x_1^3 + ax_1 + b\right)^{\frac{p-1}{2}} \left(x_1^3 + ax_1 + b\right) \mod p \\ &\equiv \left(x_1^3 + ax_1 + b\right) \mod p \end{aligned}$$

by Euler's criterion. □

So, if we are given the x coordinate of $kP = (x_1, y_1)$ for some integer k we can compute a point (x_1, y_2) such that either $(x_1, y_2) = kP$ or $(x_1, y_2) = (q-k)P$.

Lemma 2.3. *Let E be an elliptic curve over $\mathbb{Z}_p$ where $p \geq 3$ is a prime number, $P \in E(\mathbb{Z}_p)$ be a point with order q, (r, s) be a signature generated by a nonce k such that $kP = (x_1, y_1)$. If p and q have more than 2 bits, $p/q \leq 1 + \varepsilon$ and the nonces are generated randomly,*

$$P(r = x_1 \mod p) \geq 1 - 4\varepsilon.$$

Proof. The only way that we can have $r \neq x_1 \mod p$ is if $r \in [q, p-1]$. Thus, there are $p - q$ different possibilities of the x coordinate of kP such that $r \neq x_1 \mod p$. Hence, since $\mathbb{Z}_p$ is a field, there at most $2(p-q)$ possibilities of k such that $r \neq x_1 \mod p$. If we assume that the nonces are generated randomly we have

$$\begin{aligned} P(r \neq x_1 \mod p) &\leq \frac{2(p-q)}{q-2} \\ &= \frac{2(p-q)}{q}\frac{q}{q-2} \\ &\leq 4\varepsilon. \end{aligned}$$

□

For the secp256k1 curve, the curve used for bitcoin transactions, $p \equiv 3 \mod 4$ and ε can be 10^{-38}, thus, $r = x_1 \mod p$ almost surely. Therefore, given a signature with nonce k, we can compute a point which is almost surely equal to either kP or $-kP$.

2.3. Lattice attacks

2.3.1 One variable

Definition 2.2 (Lattice)**.** Given m linearly independent vectors $b_1, \ldots, b_m \in \mathbb{R}^n$, the lattice generated by them is

$$\mathcal{L}(b_1, \ldots, b_m) = \left\{ \sum_1^m k_i b_i : k_i \in \mathbb{Z} \right\}.$$

When $m = n$ we call the lattice a full-rank lattice. Throughout this article, only full rank lattices will be used. Let M be a matrix, we define $\mathcal{L}(M)$ as the lattice generated by the rows of M.

Let α be a secret integer modulo a public prime q. In the HNP an attacker is assumed to be given an oracle that given a random sequence of integers t_i, for $i\{1, \ldots, m\}$, returns a sequence a_i for $i \in \{1, \ldots, m\}$ such that $|t_i\alpha - a_i|_q \leq C$, for some $C \leq q$. That problem can be reduced to finding a solution $x_1 = b_1, \ldots, x_m = b_m, y = \alpha$ such that $|b_i| \leq C$ to the following system of equations

$$\begin{aligned} x_1 - t_1 y + a_1 &\equiv 0 \mod q, \\ &\vdots \\ x_m - t_m y + a_m &\equiv 0 \mod q. \end{aligned} \tag{2}$$

Let us a define matrix M as

$$\begin{bmatrix} q & & & & \\ & \ddots & & & \\ & & q & & \\ t_1 & & t_m & \frac{C}{q} & \\ a_1 & & a_m & & C \end{bmatrix}. \tag{3}$$

Then, we have the following theorem

Theorem 2.4. *Let $v = (b_1, \ldots, b_m, \alpha C q, -C)$ be a vector such that $x_1 = b_1, \ldots x_m = b_m, y = \alpha$ is a solution to (2). Furthermore, let us assume that $|b_i| \leq C$ for $i = 1, \ldots, m$, where C is an integer such that*

$$\log C \leq \frac{m-1}{m}\log q - \frac{m+2}{2m}\log(m+2). \tag{4}$$

Then, for small values of m, either v or $-v$ will be a vector in the LLL-reduced basis of $\mathcal{L}(M)$.

Proof. It is trivial to prove that v is a vector in $\mathcal{L}(M)$. If v is a vector in the lattice $\mathcal{L}(M)$, m is small and $\|v\| \leq \det M^{1/\dim M}$, either v or $-v$ is one of the vectors in the LLL-reduced basis [7]. Therefore, we only need to prove that

$$\|v\| \leq \left(C^2 q^{m-1}\right)^{\frac{1}{m+2}}.$$

Equivalently, we can prove that

$$\log(\|v\|) \leq \frac{2}{m+2}\log C + \frac{m-1}{m+2}\log q.$$

This can be proven using simple algebraic manipulations. □

Given the vector v, the secret α can immediately be computed. Therefore, if m is small and C and m satisfy 4, the hidden number problem can be solved using a lattice reduction algorithm.

2.3.2 Two variables

Let α and β be secret integers modulo a public prime q. In the two variables HNP an attacker is assumed to be given an oracle that given two random sequence of integers t_i, t_i' for $i\{1, \ldots, m\}$, returns a sequence a_i for $i \in \{1, \ldots, m\}$ such that $|t_i\alpha + t_i'\beta - a_i|_q \leq C$, for some $C \leq q$. That problem can be reduced to finding a solution $x_1 =$

$b_1, \ldots, x_m = b_m, y = \alpha, z = \beta$ such that $|b_i| \le C$ to the following system of equations

$$\begin{aligned} x_1 - t_1 y - t_1^{'} z + a_1 &\equiv 0 \mod q, \\ &\vdots \\ x_m - t_m y - t_m^{'} z + a_m &\equiv 0 \mod q. \end{aligned} \tag{5}$$

Let us define a matrix M as

$$\begin{bmatrix} q & & & & & \\ & \ddots & & & & \\ & & q & & & \\ t_1 & & t_m & \frac{C}{q} & & \\ t_1^{'} & & t_m^{'} & & \frac{C}{q} & \\ a_1 & & a_m & & & C \end{bmatrix}, \tag{6}$$

Theorem 2.5. *Let $v = (b_1, \ldots, b_m, \alpha Cq, \beta Cq, -C)$ be a vector such that $x_1 = b_1, \ldots x_m = b_m, y = \alpha, z = \beta$ is a solution to (5). Moreover, let us assume that $|b_i| \le C$ for $i = 1, \ldots, m$, where C is an integer such that*

$$\log C \le \frac{m-2}{m} \log q - \frac{m+3}{2m} \log(m+3). \tag{7}$$

Then, for small values of m, either v or $-v$ will be a vector in the LLL-reduced basis of $\mathcal{L}(M)$.

Proof. The proof is analogous to the proof of **Theorem 2.4**. □

Therefore, similarly to the one variable HNP, the two variables HNP can be solved using a lattice reduction algorithm.

3. Lattice attacks with one private key

For each of the following attacks let Alice be a user whose private key is y_A, with signatures $\{(r_1, s_1), \ldots, (r_n, s_n)\}$ on the messages $\{m_1, \ldots, m_n\}$.

3.1. Computing private key through small nonces

For each $i \in \{1, \ldots, n\}$ we have

$$k_i - {s_i}^{-1} r_i y_A - {s_i}^{-1} m_i \equiv 0 \mod q,$$

thus $x_1 = k_1, \ldots, x_n = k_n, y = y_A$ is a solution to the system of equations (2) where

$$\begin{cases} t_i = s_i^{-1} r_i, \\ a_i = -s_i^{-1} m_i. \end{cases}$$

Let us assume that we know a constant $C \le q$ such that $|k_i|_q \le C$ for $i = 1, \ldots, n$. If n and C satisfy (4), a lattice reduction algorithm can be used to recover the private key, according to **Theorem 2.4**. In most real life cases we do not know such constant C, so, we start by selecting a random integer $C \le q$, and using the lattice reduction algorithm on (3). If $|k_i|_q \le C$ we hope to recover the private key, otherwise the method will probably not work.

Let $m \le n$ be an integer such that m and C satisfy (4). If instead of using n signatures, we start by randomly picking m signatures and then using the lattice reduction algorithm on the matrix generated by the m signatures, the method should still work. Furthermore, the probability of success of having $|k_i|_q \le C$ for $i \in \{1, \ldots, m\}$ is at least as high as the probability of having $|k_i|_q \le C$ for $i\{1, \ldots, n\}$. Therefore, if such a value of m exists, it should be used instead of n. Moreover, the optimal value of m for C is

$$m = \min\{k : k \text{ and } C \text{ satisfy (4)}\}.$$

3.2. Computing private key through linear relations between nonces

Given $a, b \in \{1, \ldots, q-1\}$, we have

$$ak_i + bk_{i+1} - t_i y + a_i \equiv 0 \mod q$$

for each $i \in \{0, \ldots, n-1\}$, where

$$\begin{cases} t_i = as_i^{-1} r_i + bs_{i+1}^{-1} r_{i+1}, \\ a_i = -as_i^{-1} m_i - bs_{i+1}^{-1} m_{i+1}. \end{cases}$$

Thus, $x_1 = ak_1 + bk_2, x_2 = ak_2 + bk_3, \ldots, x_{n-1} = ak_{n-1} + bk_n, y = y_A$ is a solution to the system of equations (2) where the t_i and a_i are defined above.

Let us assume that we know a constant $C \le q$ such that $|ak_i + bk_{i+1}|_q \le C$ for $i \in \{1, \ldots, n-1\}$. If $n-1$ and C satisfy (4), a lattice reduction algorithm can be used to recover the private key. Similarly to what was done in 3.1, we can start by selecting $m \le n$ such that $m-1$ and C satisfy (4) and then use the lattice reduction algorithm on the matrix generated by those m signatures.

The nonces are generated with a PNRG. The attack in 3.1 computes the private key when the numbers generated by that PNRG have a disproportionately high probability of being numbers close to 0 or q. However, that does not have to be true for the PNRG. If it is known that there exists $a, b \in \{1, \ldots, q-1\}$, such that given two nonces k_1 and k_2, generated by the PNRG, there is a disproportionately high probability of $ak_1 + bk_2$ being close to 0 or q, we should use this attack. If we do not know such a and b, we will use $a = 1$ and $b = -1$.

4. Lattice attacks with two private keys

For each of the following attacks let Alice be a user whose private key is y_A, with signatures $\{(r_{A,1}, s_{A,1}), \dots, (r_{A,n_A}, s_{n_B})\}$ on the messages $\{m_{A,1}, \dots, m_{A,n_A}\}$ and Bob be a user whose private key is y_B with signatures $\{(r_{B,1}, s_{B,1}), \dots, (r_{B,n_B}, s_{B,n_B})\}$ on the messages $\{m_{B,1}, \dots, m_{B,n_B}\}$.

4.1. Computing linear equation between different private keys

We define M as $\max\{n_A, n_B\}$. We want to extend the list of signatures to M signatures for both users. In order to do that, we select the user with $min = \min\{n_A, n_B\}$ signatures and append to its list of signatures the user's first signature, and to the list of messages the user's first message. This extension, obviously, adds no new information to either of the users. However, having both list of signatures with the same size makes the subsequent attacks easier to describe and read.

For each $i \in \{1, \dots, M\}$ we have

$$\begin{cases} k_{A,i}s_{A,i}r_{B,i} - y_A r_{A,i}r_{B,i} - m_{A,i}r_{B,i} \equiv 0 \mod q, \\ k_{B,i}s_{B,i}r_{A,i} - y_B r_{A,i}r_{B,i} - m_{B,i}r_{A,i} \equiv 0 \mod q. \end{cases}$$

Thus, given $a, b \in \{1, \dots, q-1\}$, we have

$$ak_{A,i}s_{A,i}r_{B,i} + bk_{B,i}s_{B,i}r_{A,i} - yt_i + a_i \equiv 0 \bmod q$$

for each $i \in \{1, \dots, M\}$ where

$$\begin{cases} y = ay_A + by_B, \\ t_i = r_{A,i}r_{B,i}, \\ a_i = -am_{A,i}r_{B,i} - bm_{B,i}r_{A,i}. \end{cases}$$

Hence, $x_i = ak_{A,i}s_{A,i}r_{B,i} + bk_{B,i}s_{B,i}r_{A,i}, y = ay_A + by_B$ for $i = 1, \dots, M$ is a solution to (2), where the t_i and a_i are the ones defined above.

If there is a known constant C such that C and M satisfy (2), and $|ak_{A,i}s_{A,i}r_{B,i} + bk_{B,i}s_{B,i}r_{A,i}|_q \le C$ for $i = 1, \dots, M$, we can compute $ay_A + by_B \mod q$ using a lattice reduction algorithm, so we get an equation

$$ay_A + by_B \equiv c \mod q, \quad a, b \not\equiv 0 \mod q.$$

That is not enough to recover either y_A or y_B. Nonetheless, that information can still be used to recover the private keys.

One way is to use the method described but instead of using $\{a, b\}$ in (4.1), using $\{a', b'\}$ such that $a'a^{-1} \not\equiv b'b^{-1} \mod q$. If the attack is successful and returns an equation $a'y_A + b'y_B \equiv c' \mod q$ we have

$$\begin{cases} y_A \equiv (ab' - a'b)^{-1}(b'c - bc') \mod q, \\ y_B \equiv (ab' - a'b)^{-1}(a'c - ac') \mod q, \end{cases}$$

therefore, we can recover both private keys.

Similarly to what was done in the previous attacks, we can start by selecting $m \le M$ such that m and C satisfy (4), selecting m signatures and then using the lattice reduction algorithm on the matrix generated by those m equations.

As mentioned before, if we can use the method and compute two different linear equations for y_A and y_B, we can recover both y_A and y_B. However, computing one linear equation can be useful even if we can not compute another linear equation between the private keys. In Bob's equations, we can replace y_B by $b^{-1}(c - ay_A) \mod q$. After doing it, we get $n_A + n_B$ equations for y_A when before we only had n_A. Then, we can use one of the previous attacks to the $n_A + n_B$ equations. This is especially useful if the attack returns a linear equation between a user with a small amount of signatures and one with a large amount. In real world applications there are going to be many user which can have as little as one signature and there are also going to be users with millions of signatures. If this method can find a linear equation between the private keys of these two different types of users, the user with the small amount of signatures gets a great increase on the number of equations for its private key.

If we have a user Charlie, whose private key is y_C and we can compute one linear equation between y_A and y_B, one between y_A and y_C and one between y_B and y_C and all the linear equations are independent, we can recover y_A, y_B and y_C. Therefore, we can compute y_A, y_B, y_C without having two different linear equations between any of the two private keys. This method can be extended to as many users as we want to. That is, we can almost surely recover $y_1, \dots y_n$. if we can find a system of equations

$$\begin{cases} a_1y_1 + b_1y_2 \equiv c_1 \mod q, \\ \quad\vdots \\ a_ny_n + b_ny_1 \equiv c_n \mod q. \end{cases}$$

4.2. Computing two private keys through linear relation between nonces

In order to recover Alice's and Bob's private keys using the attack in 4.1 we need two successful lattice reduction algorithms. Moreover, we need to have $|ak_{A,i}s_{A,i}r_{B,i} + bk_{B,i}s_{B,i}r_{A,i}|_q \le C$ in order for the algorithm to be successful. Thus, the values which we need to be small depend on $s_{A,i}$ and $s_{B,i}$ which behave as random variables, hence, the probability of the algorithm being successful is not higher when a bad PRNG is used to generate the nonces which is another problem with the attack. In order to try to fix both these problems this new method is implemented.

Let us start by assuming that $n_A = n_B = n$. Then, given $a, b \in \{1, \dots, q-1\}$, we have

$$\begin{cases} ak_{A,i} - as_{A,i}^{-1} r_{A,i} y_A - as_{A,i}^{-1} m_{A,i} \equiv 0 \mod q, \\ bk_{B,i} - bs_{B,i}^{-1} r_{B,i} y_B - bs_{B,i}^{-1} m_{B,i} \equiv 0 \mod q, \end{cases}$$

for each $i \in \{1, \dots, n\}$, thus

$$ak_{A,i} + bk_{B,i} - t_i y - t_i^{'} z + a_i \equiv 0 \mod q.$$

Therefore, $x_i = ak_{A,i} + bk_{B,i}, y = y_A, z = y_B$ is a solution to (5) where

$$\begin{cases} t_i = as_{A,i}^{-1} r_{A,i}, \\ t_i^{'} = bs_{B,i}^{-1} r_{B,i} y_B, \\ a_i = -as_{A,i}^{-1} m_{A,i} - bs_{B,i}^{-1} m_{B,i}. \end{cases} .$$

Let us now assume that Alice is a user with n_A signatures and Bob is a user with n_B signatures such that $n_B > n_A$. Similarly to what was done in 4.1 we want to extend Alice's list of signatures to n_B signatures. For that attack, n_A could be as small as one and it still could be enough to successfully use the lattice attack. That was possible because the randomness of the t_i depended on $r_{A,i} r_{B,i}$, hence, as long as the sequence $r_{B,i}$ was random, the sequence $r_{A,i}$ did not have to be random. In this method if only one signature was used for Alice, t_i would be constant, thus there would be an integer x such that $|t_i x - t_i y_A|_p \leq 1$ for $i \in \{1, \dots, n_B\}$, which, in turn, means that if $(b_1, \dots, b_{n_B}, y_A, y_B, 1)M$ is a small vector in the lattice generated by M the vector $(b_1, \dots, b_{n_B}, x, y_B, 1)M$ will also be a small vector in the lattice generated by M. Thus, the algorithm might return y_B but the probability of it returning y_A is negligible. Therefore, n_A can not be as small as we want. As far as we know there are no known inequations relating n_A and C, but we expect that if n_A and C satisfy (4) the attack is successful.

In a lattice reduction we want t_i to behave as randomly as possible, thus, instead of only repeating one signature as it was done in 4.1 we will repeat the maximum possible number of signatures. In order to achieve that, we append her list of signatures to itself $\lfloor n_B n_A \rfloor$ times and then take its first n_B entries and then use the attack on the corresponding equations.

Similarly to what was done in the previous attacks, we can start by selecting $m \leq n_B$ such that m and C satisfy (7), selecting m equations and then using the lattice reduction algorithm on the matrix generated by those m equations.

If it is known that there exist $a, b \in \{1, \dots, q-1\}$, such that given two nonces k_1 and k_2, generated by the PNRG, there is a disproportionately high probability of $ak_1 + bk_2$ being close to 0 or q, this attack should be used. If we do not know such a and b, we will try to use the method with $a = 1$ and $b = -1$.

5. Results

We wish to compute the probability of success of the lattice attacks as a function of m and C, where m is the number of equations used, and C is a constant such that $|b_i|_q \leq C$ for $i = 1, \dots, m$.

It was proved in the previous sections that the attacks are reducible to solutions to the HNP. Thus, in order to test the attacks, we will test the attacks in **2.3.1** and **2.3.2**. We will fix C and compute the probability of success as a function of m. This test is useful, for example, to find out how many equations should be used if there is a side-channel attack which can recover a number of most, or least, significant bits.

The prime number q used in these tests will be the order of P in the elliptic curve secp256k1, the curve used in Bitcoin's public key-cryptography. The numbers t_i and b_i will be generated with the RandomInteger function in Mathematica and the lattice reduction algorithm used is also the one defined in Mathematica. These test were run in a computer with 8 GB RAM and a i7-7500U processor.

5.1. One variable lattice attack

Given C and m we want to compute the probability of the one variable hidden number problem being successful. In order to do that, we design an algorithm which starts by randomly generating an integer $1 \leq \alpha \leq q-1$, the vector $(t_1, \dots, t_m)$, where $1 \leq t_i \leq q-1$, and the vector $(b_1, \dots, b_m)$ where $-C \leq b_i \leq C$. Then, it computes $(a_1, \dots, a_m)$, in such a way that $x_i = b_i, y = \alpha$ is a solution to (2). Lastly, the algorithm uses LLL on $\mathcal{L}(M)$ where M is the matrix (3), and it returns 1 if either the vector $v = (b_1, \dots, b_m, \alpha C/q, -C)$ or $-v$ is a vector in the reduced basis and 0 otherwise. Each of these tests were run thirty times. The average of those thirty tests was taken, and that average is the experimental probability of success.

As seen in Fig.1 the results for $C = 2^{249}$ are very poor. For most number of equations tested the probability of success is equal to zero, and it is smaller than 7% for all number of equations tested. We could not run the tests for values of m greater than 90, since the tests were already too slow for

$m = 90$, so we are not sure if this is a problem which could be corrected if more equations were used, or if for $C = 2^{249}$ the attack can not recover the private key with a high probability of success. For $C = 2^{248}$, if $m \geq 59$, the probability of success is at least 45%. The probability of success does not converge as m grows, however, it is enough to greatly endanger the security of the private keys if there exists a side-channel attack which can recover 7 MSB or LSB of the nonces. For $C = 2^{247}$, if $m \geq 42$ there is at least a 70% probability of success and for $m \geq 45$ there is at least a 90% probability of success. For $C = 2^{246}$, if $m \geq 36$ there is at least a 85% probability of success and for $m \geq 38$ there is a 100% probability of success. For $C = 2^{245}$, if $m \geq 31$ there is at least a 90% probability of success. As expected, for $C \in \{2^{245}, 2^{246}, 2^{247}\}$ the attack has a higher probability of success than for $C \in \{2^{248}, 2^{249}\}$. Moreover, for each $C \in \{2^{245}, 2^{246}, 2^{247}\}$ a number of equations can be chosen such that the probability of success of the attack is at least 90%.

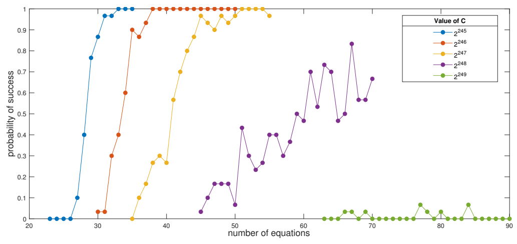

Figure 1: Probability of success as function of the number of equations for the one variable HNP.

5.2. Two variables lattice attack

Given C and m we want to compute the probability of the two variables lattice attack computing the private keys. In order to do that we design an algorithm which starts by randomly generating two integers $1 \leq \alpha, \beta \leq q-1$, the vectors $(t_1, \ldots, t_m)$ and $(t_1^{'}, \ldots, t_m^{'})$, where $1 \leq t_i, t_i^{'} \leq q-1$, and the vector $(b_1, \ldots, b_m)$ where $-C \leq b_i \leq C$. Then, it computes $(a_1, \ldots, a_m)$, in such a way that $x_i = b_i, y = \alpha, z = \beta$ is a solution to (5). Lastly, the algorithm uses LLL on $\mathcal{L}(M)$ where M is the matrix (6), and it returns 1 if either the vector $v = (b_1, \ldots, b_m, \alpha C/q, \beta C/q, -C)$ or $-v$ is a vector in the reduced basis and 0 otherwise. Each of these tests were run thirty times. The average of those thirty tests was taken, and that average is the experimental probability of success.

As it can be seen in Fig. 2 for $C = 2^{245}$ the probability of success does not seem to converge as m grows, moreover the probability of success is smaller than 60% for all values of m tested. These poor results are expected since the number of equations used are close to 100. For $C = 2^{244}$, the probability of success as a function of m is not an increasing function and it does not seem to converge as m grows . However, for $m \geq 66$ the probability of success is greater than 70% and for $m \geq 73$ it is greater than 80%. Thus, for a large enough number of equations, we have a high probability of success of recovering α and β assuming that $|b_i|_q \leq 2^{244}$. For each other value of C tested, there is an integer n_C, such that for $m \geq n_C$, the probability of success for m equations is equal to 1. Therefore, for those values of C the two variables lattice attack will recover both private keys.

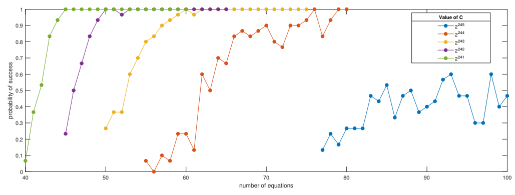

Figure 2: Probability of success as function of the number of equations for the two variables HNP.

5.3. Comparing the attacks when a user has a random solution to the equation

As it was described in 4.2 the number of equations used for Alice and Bob do not always have to be the same. Let us assume that Alice has n signatures and Bob has m signatures, where $n < m$. For each C, we will fix m and compute the probability of success of the two variables lattice attack as a function of n. Moreover, we will also compute the probability of success for the one variable lattice attack for the same C and n. Then, we will compare the results of both methods. The values of m used for each C are presented in Table 1.

C	2^{245}	2^{244}	2^{143}	2^{242}
m	93	79	62	53

Table 1: Experimental optimal values of m as a function of C, for the two variables lattice attack.

The test results are presented in the graphics below.

For all values of C tested the probability of success for the one variable method converges to 1 as n grows. The same does not happen for the two variables method. For $C = 2^{245}$ and $C = 2^{244}$, the probability of success for this method does not converge to 1 as n grows, in fact, for the largest values of n tested, there is a decrease in the probability of success, this is most noticeable for $C = 2^{245}$.

For all values of C, there is at least one value n where the probability of success for the two variables lattice attack is higher than the probability

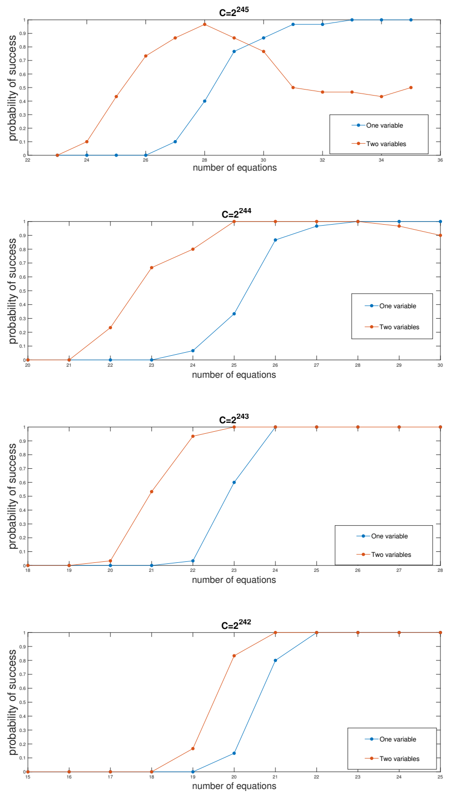

Figure 3: Comparison between the one and two variables lattice attack.

of success for the one variable lattice attack. Even though the two variables lattice attack has a better probability of success than the one variable lattice attack, the two variables lattice attack needs to have m inequalities $|b_i|_q \leq C$, while the one variable lattice attack only needs to have n inequalities $|b_i|_q \leq C$.

5.4. User whose solution to the equation is zeros

In order to compare the different attacks when we only use n inequalities in both methods let us assume that we have two vectors $t = (t_1, \dots, t_n)$ and $a = (a_1, \dots, a_n)$. Suppose that the system of equations

$$x_i - t_i y + a_i \equiv 0 \mod q, \quad \text{for } i = 1, \dots, n,$$

is satisfied by $x_1 = b_1, \dots, x_n = b_n, y = \alpha$, where $|b_i|_q \leq C$ for $i \in \{1, \dots, n\}$. Let us define a vector b as $(b_1, \dots, b_n)$. Similarly to what was done in the previous section, t is appended with itself $\left\lfloor \frac{m}{n} \right\rfloor$ and the m first entries are taken and t is updated to be this vector. We do the same for the vectors a and b. For this $t = (t_1, \dots, t_m)$ and $a = (a_1, \dots, a_m)$ we have the system of equations

$$x_i - t_i y + a_i \equiv 0 \mod q, \quad \text{for } i = 1, \dots, m$$

which has $x_1 = b_1, \dots, x_m = b_m, y = \alpha$ as a solution.

To use the two variables method, let us start by randomly generating a vector $t' = (t_1^{'}, \dots, t_m^{'})$ and an integer β. Then, we compute the vector $a^{'} = (a_1^{'}, \dots, a_m^{'})$ in such a way that $x_1 = 0, \dots, x_m = 0, z = \beta$ is a solution to

$$x_i - t_i^{'} z + a_i^{'} \equiv 0 \text{ mod } q, \quad \text{for } i = 1, \dots, m$$

Adding (5.4) with (5.4) we have the system of equations

$$x_i - t_i y - t_i^{'} z + a_i^{'} + a_i \equiv 0 \text{ mod } q, \quad \text{for } i = 1, \dots, m,$$

which is satisfied by $x_1 = b_1, \dots, x_m = b_m, y = \alpha, z = \beta$.

This is now a lattice problem in two variables with a solution where $|b_i|_q \leq C$ for $i \in \{1, \dots, m\}$. However, this problem is not equivalent to the one in 5.3, since this solution has repeated values of b_i and in that section the values of b_i were random in the interval $\{-C, \dots, C\}$. The results for this method are presented in the graphics below.

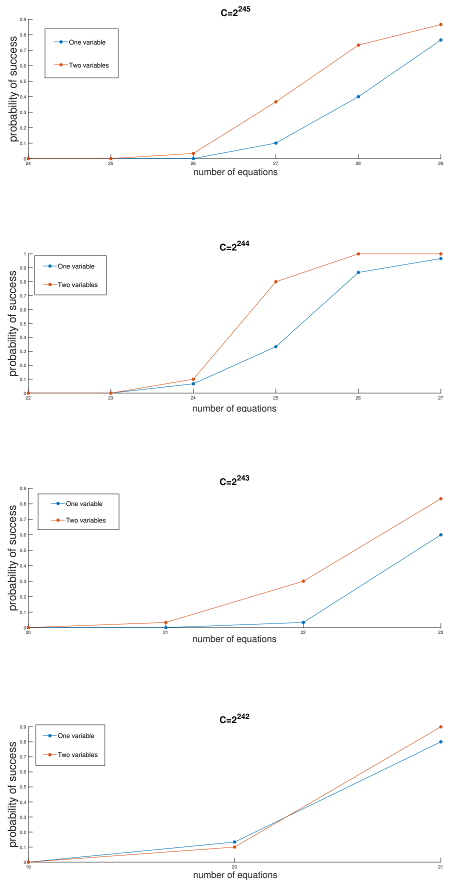

Figure 4: Comparison between the one and two variables lattice attack.

The probability of success for this version of the two variables method is, as expected, not as high as the one in 5.3. However, surprisingly, for most values of C and n the method has a higher probability of success than the one variable method. The increase in performance is especially evident for $C \in \{2^{245}, 2^{244}, 2^{243}\}$. For those values of C, the probability of success for the two variables lattice attack is never lower than the probability of success for the one variable lattice attack. Furthermore, the only values of n where the probability of success for the two variables lattice attack is not higher than the probability of success for the one variable lattice attack is when the probabilities are both equal to zero. Despite the fact that the probability of success for the two variables attack is better than the probability of success for the one variable attack, whenever the probability of success is 0 for the one variable method the probability of success for the two variables method is low, it is never more than 10%. Therefore, unlike the method in 5.3, in this method if the private key can not be computed with the one variable lattice attack it can only be computed with a low probability with the two variables lattice attack. Notwithstanding, if the one variable method was not able to compute the private key it is possible that this attack can and, consequently, we should try to use this method for that user.

6. Conclusions

The results show that for this solution to the HNP and using the LLL algorithm implemented in Mathematica, converting a one variable lattice attack to a two variables lattice attack can increase the probability of success. We also confirmed that the ECDSA is secure against all these attacks if the nonces are randomly generated. However, if the nonces are generated with some weak PNRG, it may be possible to design an attack which can recover the private key. Moreover, if one is able to design a side-channel attack which recovers a number of most, or least, significant bits, the private key can also be recovered with a high probability.

6.1. Future Work

For our tests we needed to have at least 7 bits leaked to have a good probability of success. It is possible that if we had tested for lattices with higher dimensions we would be able to recover the private key with only 6 leaks, but for higher dimensions than the ones tested the attacks took too much time and we were not able to compute them. So, one future test that could be done would be to run these tests with a faster programming languages. Another change that could be done is using the BKZ 2.0 [8] as the lattice reduction algorithm instead of the LLL, since the BKZ 2.0 is a better lattice reduction algorithm for high dimensions. It would also be interesting to compare the two variables lattice attack and the one variable lattice attack using a faster programming language and the BKZ 2.0 for larger values of C and verify if the two variables lattice attack still has a higher probability of success than the one variable lattice attack.

Another attack which could be implemented is one attack which uses a group attack and a lattice attack, it could start by using a group attack and if the group attack does not recover the private key, using that information to design the lattice attack. Other attacks which could be designed is to try to use a three or more variables lattice attack and ascertaining if that attack increases the probability of success, such as the two variables lattice attack did.

References

[1] ANSI X9.62:2005: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), 2005.

[2] L. Babai. On Lovsz lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(No. 1), 1986.

[3] N. Benger, J. van de Pol, N. P. Smart, and Y. Yarom. "Ooh Aah... Just a Little Bit" : A Small Amount of Side Channel Can Go a Long Way. In *Cryptographic Hardware and Embedded Systems – CHES 2014*, pages 75–92. Springer Berlin Heidelberg, 2014.

[4] D. Bleichenbacher. On The Generation of One-Time Keys in DL Signature Schemes. In *Presentation at IEEE P1363 Working Group meeting*, November 2000.

[5] D. Bleichenbacher. On the Generation of DSA One-Time Keys. In *Presentation at Cryptography Research, Inc., San Francisco, CA*, 2007.

[6] D. Boneh and R. Venkatesan. Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes. *In: Koblitz N. (eds) Advances in Cryptology CRYPTO 96. CRYPTO 1996. Lecture Notes in Computer Science*, vol 1109. Springer, Berlin, Heidelberg, 1996.

[7] J. Breitner and N. Heninger. Biased nonce sense: Lattice attacks against weak ecdsa signatures in cryptocurrencies. Cryptology ePrint Archive, Report 2019/023, 2019. https://eprint.iacr.org/2019/023.

[8] Y. Chen and P. Q. . Nguyen. BKZ 2.0: Better Lattice Security Estimates. In *Proceedings*

of the 17th International Conference on The Theory and Application of Cryptology and Information Security, ASIACRYPT'11, pages 1–20, Berlin, Heidelberg, 2011. Springer-Verlag.

[9] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22(No. 6):644–654, Nov 1976.

[10] N. A. Howgrave-Graham and N. Smart. Lattice Attacks on Digital Signature Schemes. *Designs, Codes and Cryptography*, 23:283–290, 2001.

[11] A. K. Lenstra, H. W. L. Jr., and L. Lovsz. Factoring Polynomials with Rational Coefficients". *Mathematische Annalen*, Vol. 261(No. 4):515–534, 1982.

[12] E. D. Mulder, M. Hutter, M. E. Marson, and P. Pearson. Using Bleichenbachers Solution to the HiddenNumber Problem to Attack Nonce Leaks in384-Bit ECDSA. *J Cryptogr Eng*, 4(33):201–217.

[13] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[14] P. Q. Nguyen and I. E. Shparlinski. The Insecurity of the Elliptic Curve Digital Signature Algorithm with Partially Known Nonces. *Designs, Codes and Cryptography*, 30:201–217, 2003.

[15] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66(2):181–199, Sept. 1994.