DiSSECT: Distinguisher of Standard & Simulated Elliptic Curves via Traits

Vladimir Sedlacek[1,2], Vojtech Suchanek[1], Antonin Dufka[1], Marek Sys[1], and Vashek Matyas[1]

[1] Masaryk University, Czech Republic
[2] Université de Picardie Jules Verne, France
{vlada.sedlacek, vojtechsu, dufkan, syso, matyas}@mail.muni.cz

Abstract. It can be tricky to trust elliptic curves standardized in a non-transparent way. To rectify this, we propose a systematic methodology for analyzing curves and statistically comparing them to the expected values of a large number of generic curves with the aim of identifying any deviations in the standard curves.
For this purpose, we put together the largest publicly available database of standard curves. To identify unexpected properties of standard generation methods and curves, we simulate over 250 000 curves by mimicking the generation process of four standards. We compute 22 different properties of curves and analyze them with automated methods to pinpoint deviations in standard curves, pointing to possible weaknesses.

Keywords: elliptic curves · standards · simulations · testing tool.

1 Introduction

Many prominent cryptographers [Sch13; Loc+14; Ber+15; CLN15; Sco99] criticize the selection of curves in all major elliptic curve standards [ANS05; Cer10; Age11; SSL14]. The lack of explanation of parameters used in cryptographic standards provides a potential space for weaknesses or inserted vulnerabilities.

This is a serious matter, considering that the popular NIST curves have been designed by the NSA, which has been involved in a number of incidents: the Dual EC DRBG standard manipulation [Hal13; BLN16; Che+], Clipper chip backdoor [Tor94], or the deliberate weakening of the A5 cipher [BD00]. Bernstein et al. [Ber+15] justify such skepticism, showing that the degrees of freedom in the generation of elliptic curves offer a means to insert a backdoor. Relying on the supposed implausibility [KM16] of weak curves escaping detection for such a long time may prove catastrophic. As the usage of NIST curves increases with time [Val+18], the impact of any vulnerability would be extensive at this moment.

Even though newer, more rigidly generated curves like Curve25519 [Ber06], Ed448-Goldilocks [Ham15b] or NUMS curves [Bla+14] are on the rise, Lochter et al. [Loc+14] argue that *"perfect rigidity, i.e., defining a process that is accepted as completely transparent and traceable by everyone, seems to be impossible."*

Thus, a thorough wide-scale analysis of the standard curves is important to establish trust in elliptic curve cryptosystems.

There is no clear way of looking for unknown vulnerabilities, especially within elliptic curve cryptosystems with such a rich and complex theory behind them. Our main idea is that if a hidden weakness is present in a curve, it can manifest itself via a statistical deviation when we compare the curve to a large number of generic curves. We consider two cases: either the deviation stems from the generation method of the curve, or from the choice of its parameters. Consequently, we compare the curve to two types of curves – random curves in the former case and curves generated according to the same standard in the latter case.

Core contributions. Our work provides the following contributions:

- We assemble the first public database of all standard[3] curves (to the best of our knowledge), with comprehensive parameter details.
- We develop an open-source, extensible framework DiSSECT for generating curves according to known standards, and for statistical analysis of the standards. This includes a large number of test functions and visualization tools.
- We find properties of the standard GOST curves [PLK06; SF16] that are inconsistent with the claimed [ANS18] generation method and present unreported properties of the BLS12-381 [BLS02] curve.

In addition, we offer a new methodology for testing properties and potential weaknesses of standard curve parameters and their generation methods. By a systematic analysis of standard curves, we rule out multiple types of problems, raising the level of trust in most of the curves.

This paper is organized as follows: In Section 2, we briefly survey the major elliptic curve standards. Section 3 introduces our database and explains our methodology for simulating and distinguishing curves. Section 4 gives an overview of our proposed trait functions and presents the findings of the outlier detection. We report on the technicalities of our tool in Section 5 and draw conclusions in Section 6. Appendix A contains descriptions of individual traits.

2 Background

Throughout the paper, we will use the following notation: $p \geq 5$ is a prime number, $\mathbb{F}_p$ is a finite field of size p, $E(a,b)\colon y^2 = x^3 + ax + b$ is an elliptic curve over $\mathbb{F}_p$, $n = \#E(\mathbb{F}_p)$ is the order of the group of points over $\mathbb{F}_p$, l the largest prime factor of n, $h = n/l$ is the cofactor, and $t = p + 1 - n$ is the trace of Frobenius of E. Parts of this section are adopted from [Sed22].

Even though pairing-friendly curves and curves over binary and extension fields have found some applications, we focus mainly[4] on the prevalent category

[3] In this work, we use the adjective "standard" for widely used curves (from actual standards and even that are/were a de facto standard).

[4] However, binary, extension and pairing-friendly curves are included in our database as well and have been included partly in our analysis.

of prime field curves that are recommended for classical elliptic curve cryptosystems. We include Montgomery and (twisted) Edwards curves as well, though for unification purposes, we convert them to the short Weierstrass form, which is universal.

Known curve vulnerabilities. When using elliptic curve cryptography (ECC) in the real world, both sides of the scheme must agree on a choice of a particular curve. We need curves where the elliptic curve discrete logarithm problem (ECDLP) is difficult enough. The SafeCurves website [BL] presents a good overview of known curve vulnerabilities (and discusses implementation-specific vulnerabilities as well). In short, to avoid the known mathematical attacks, curve should satisfy the following criteria:

- p should be large enough, e.g., over 256 bits if we aim for 128-bit security;
- l should be large enough, as this determines the complexity of the Pohlig-Hellman attack [PH78] (again, roughly 256 bits for 128-bit security); ideally, the same should hold for the quadratic twist of the curve (to avoid attacks on implementations[5]);
- the curve should not be anomalous (i.e., $n \neq p$), otherwise the Semaev-Satoh-Araki-Smart additive transfer attack applies [Sma99; Sem98; SA+98];
- the embedding degree of E (i.e., the order of p in $\mathbb{F}_l^\times$) should be large enough (e.g., at least 20), otherwise the MOV attack based on multiplicative transfer using the Weil and Tate pairings applies [Sem96; FR94; MOV93] – in particular, this rules out the cases $t = 0$ and $t = 1$;
- the absolute value of the CM field discriminant (i.e., the absolute discriminant of the field $\mathbb{Q}(\sqrt{t^2 - 4p})$ should not be too low ([BL] suggests at least 2^{100}), otherwise Pollard's ρ attack can be speeded up due to the presence of efficiently computable endomorphisms of the curve [GLV01]. This does not pose a serious threat for the moment though, as the limits of the speedup are reasonably well understood.

2.1 Overview of standard curve generation

The main approach for finding a curve that satisfies the mentioned conditions is repeatedly selecting curve parameters until the conditions hold. The bottleneck here is point-counting: we can use the polynomial SEA algorithm [Sch95], but in practice, it is not fast enough to allow on-the-fly ECC parameter generation.

Several standardization organizations have therefore proposed elliptic curve parameters for public use. To choose a curve, one has to first pick the appropriate base field. Several standards also choose primes of special form for more efficient arithmetic, e.g., generalized Mersenne primes [Sol11] or Montgomery-friendly primes [CLN15]. When the field and the curve form are fixed, it remains to pick two[6] coefficients, though one of them is often fixed.

[5] This is often ignored in the standards though, except for the NUMS standard and individual curves such as Curve25519.

[6] At least for short Weierstrass, Montgomery and twisted Edwards curves.

In this subsection, we survey the origin of all the standard curves. For easier orientation, we divide the generation methods of standard curves into three rough categories. Curves of unknown or ambiguous origin are hard to trust, while verifiably pseudorandom ones make use of one-way functions in hopes of addressing this. Other rigid methods go even further in attempts to increase their transparency.

Unknown or ambiguous origin. The following is a list of curves we analyze and whose method of generations is either completely unexplained or ambiguous.

- The 256-bit curve `FRP256v1` has been recommended in 2011 by the French National Cybersecurity Agency (ANSSI) in the Official Journal of the French Republic [Jou11]. The document does not specify any method of generation.
- The Chinese SM2 standard [SSL14] was published in 2010 [DY15] by the Office of State Commercial Cryptography Administration (OSCCA). The standard recommends one 256-bit curve but does not specify the generation method.
- In addition to the Russian GOST R 34.10 standard from 2001, six standardized curves were provided in [PLK06] and [SF16]. Again, no method of generation was specified in the original standard, although there were some attempts for explanation afterward [ANS18]. We discuss this in further sections.
- The Wireless Application Protocol Wireless Transport Layer Security Specification [Wir00] recommends 8 curves, 3 of which are over prime fields and are not copied from previous standards. Their method of generation is not described.
- Apart from verifiably pseudorandom curves, the Standards for Efficient Cryptography Group (SECG) [Cer] recommends so-called Koblitz curves[7], which possess an efficiently computable endomorphism. However, the standard only vaguely states *"The recommended parameters associated with a Koblitz curve were chosen by repeatedly selecting parameters admitting an efficiently computable endomorphism until a prime order curve was found."*

Verifiably pseudorandom curves. Aiming to re-establish the trust in ECC, several standards have picked the coefficients a, b in the Weierstrass form $y^2 = x^3 + ax + b$ in a so-called *verifiably pseudorandom* way, as Figure 1 demonstrates.

The details differ and we study them more closely in Section 3.2. The common idea is that the seed is made public, which limits the curve designer's freedom to manipulate the curve. However, this might not be sufficient, as Bernstein et al. [Ber+15] show that one could still iterate over many seeds (and potentially also over other "natural" choices) until they find a suitable curve. If a large enough proportion of curves (say, one in a million) is vulnerable to an attack known to the designer, but not publicly, this offers an opportunity to insert a backdoor. In particular, we should be suspicious of curves whose seed's origin is unknown.

[7] Analyzing these curves is of great importance due to the usage of `secp256k1` in Bitcoin [Nak08].

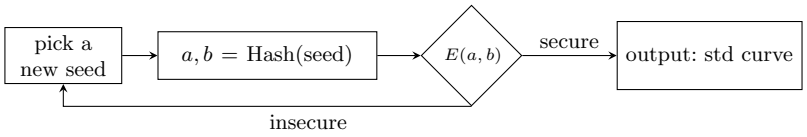

Fig. 1: A simplified template for generating verifiably pseudorandom curves over a fixed field $\mathbb{F}_p$.

In our analysis, we will focus on two major standards containing verifiably pseudorandom generation procedures:

- The Brainpool curves, proposed in 2005 by Manfred Lochter and Johannes Merkle under the titles "ECC Brainpool Standard Curves and Curve Generation" [Brainpool] to focus on issues that have not been previously addressed such as verifiable choice of seed or usage of prime of nonspecial form.
- NIST in collaboration with NSA presented the first standardization of curves was in FIPS 186-2 in 2000 [Nat00]. However, the used generation method already appeared in ANSI X9.62 in 1998 [Com+98]. Other standards recommend these curves (e.g., SECG [Cer] or WTLS [Wir00]).

Another notable paper considering the verifiably pseudorandom approach is "The Million dollar curve" which proposed a new source of public entropy for the seeds, combining lotteries from several different countries [Bai+15].

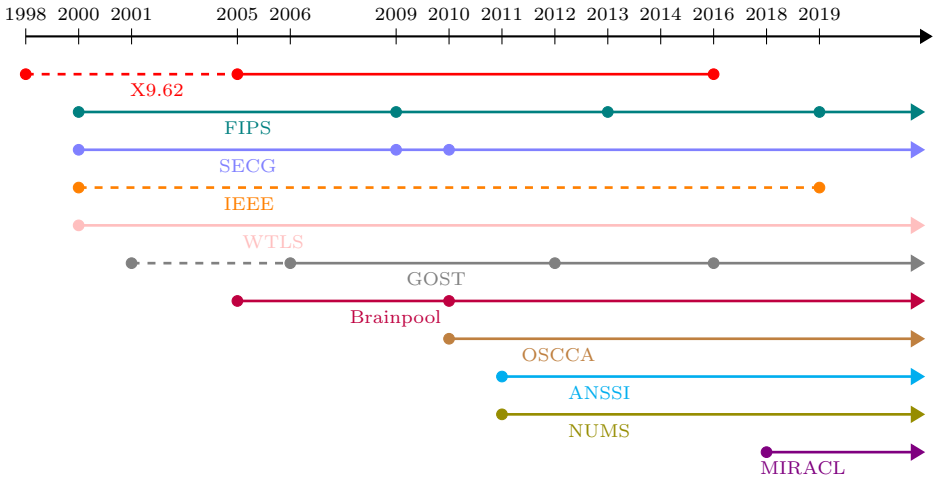

Fig. 2: Timeline of standardization of publicly available prime-field curves. Dashed line indicates that the source does not specify curve parameters. Individual curves (e.g., Curve25519) are omitted.

Other rigid methods. As a reaction to the NIST standard curves, an interest in faster curves generated using rigid and verifiable methods has emerged:

- Three Weierstrass curves and three twisted Edwards curves were proposed by Black et al. [Bla+14] as an Internet draft called "Elliptic Curve Cryptography (ECC) Nothing Up My Sleeve (NUMS) Curves and Curve Generation" [Bla+14].
- MIRACL library [MIR18] combines approaches of NUMS and Brainpool by extracting seeds from well-known constants and iteratively incrementing one of the parameters of a curve in the Weierstrass form.
- Bernstein's Curve25519 and its sibling Ed25519 were created in 2006 [Ber06] and since then have been widely accepted by the cryptography community. The curve is in the Montgomery form and allows extremely fast x-coordinate point operations while meeting the SafeCurves security requirements. In 2013, Bernstein et al. [Ber+13] proposed the curve Curve1174 with an encoding for points as strings indistinguishable from uniform random strings (the Elligator map).
- To address higher security levels and maintain good performance, several authors developed curves of sizes in the 400-521 bit range such as the Edwards curve Curve41417 by Bernstein, Chuengsatiansup, and Lange [BCL14], Ed448-Goldilocks by Hamburg [Ham15a], E-3363 by Scott [Sco15] or M, E curves by Aranha1 et al. [Ara+13].

3 Methodology

Our ultimate goal is to enable assessment of security for standard curves. But since it is not clear how to look for unknown curve vulnerabilities, we now aim to pinpoint possible problems via identifying standard curves that deviate from other curves in specific aspects. To find deviations we employ one of the following strategies:

1. **Compare curves from a standard to corresponding simulated ones.** If the standard curves were generated using the defined processes without any hidden conditions, it would be unexpected to statistically distinguish them from our simulated ones, yet we try to do exactly that. We are looking for a standard curve achieving a value that is highly improbable when compared to the simulated curves.
2. **Compare curves from a standard to Random curves.** If the generation algorithm introduces a systematic bias, the first strategy will not allow us to find it – the problem might occur in both the standard and simulated curves. Thus this strategy tries to detect any sort of unexpected behaviour by comparing curves from a given standard to *Random*[8] curves.

[8] In the remainder of this work, the word Random with capital R refers to the curves we generated by our own method to be as generic as possible.

To target specific curve properties, we describe and implement traits, which are functions that take a curve as an input (sometimes with additional parameters) and output numerical results. We run these traits on standard curves as well as simulated ones whenever it is computationally feasible and store the results in our database.

3.1 Standard curve database

DiSSECT is, as far as we know, the first public database of all standard elliptic curves, divided into 18 curve categories by their source. The database includes:

1. verifiably pseudorandom curves (X9.62, NIST, SEC, Brainpool);
2. pairing-friendly curves: Barreto-Lynn-Scott curves [BLS02], Barreto-Naehrig curves [Per+11], Miyaji-Nakabayashi-Takano curves [MNT01]);
3. amicable curves: Tweedledee/Tweedledum [BGH19], Pallas/Vesta [Hop20];
4. rigidly generated NUMS curves and curves from the MIRACL library [MIR18];
5. Bernstein's high performance curves [Ber06] and M, E curves [Ara+13];
6. curves from the standards ANSSI [Jou11], OSCCA [SSL14], GOST [PLK06; SF16], OAKLEY [Orm98], WTLS [Wir00], ISO/IEC [ISO17] and others;

Although our analysis focuses mainly on prime-field curves, the database contains 31 curves over binary fields and one over an extension field. Currently, there are 188 standard curves in total. Note that we also include curves that were but are no longer supported by the standards, and curves that are not recommended for public use, but have been included in the documents for various reasons (e.g., curves for implementation checks). The database provides filtering by bit-length, field type, cofactor size, and curve form.

Additionally, our database contains five categories of simulated curves: $\text{X9.62}_{\mathsf{sim}}$, $\text{Brainpool}_{\mathsf{sim}}$, $\text{NUMS}_{\mathsf{sim}}$, $\text{Curve25519}_{\mathsf{sim}}$ and Random[9].

For each curve, we also precomputed usual properties such as the CM discriminant, the j-invariant, the trace of Frobenius t, and the embedding degree. This precomputation significantly speeds up computation of some traits.

Source	#	**Source**	#	**Source**	#	**Source**	#
X9.62	40	BARP	6	OSCCA	1	$\text{X9.62}_{\mathsf{sim}}$	120k
Brainpool	14	BLS	6	BN	16	$\text{NUMS}_{\mathsf{sim}}$	1.2k
NUMS	24	GOST	9	AMIC	4	$\text{Curve25519}_{\mathsf{sim}}$	784
SECG	33	ISO	4	DJB	10	$\text{Brainpool}_{\mathsf{sim}}$	12k
NIST	15	MNT	10	ANSSI	1	Random	250k
MIRACL	8	OAKLEY	2	WTLS	8	other	11

Table 1: Numbers of elliptic curves in our database grouped by their source.

[9] The set of Random curves and their trait results is currently still evolving.

3.2 Simulations

We have picked four major standards X9.62 [Com+98], Brainpool [Brainpool], NUMS [Bla+14] and Curve25519 [Ber06] for simulations, since their generation method was explained in enough detail and can be easily extended for large scale generation. At a few points, the standards were a little ambiguous, so we filled the gaps to reflect the choices made for the actual standard curves whenever possible. We have simulated over 120 000 X9.62 curves, 12 000 Brainpool curves, 1 200 NUMS curves and 750 Curve25519 curves[10].

The aim of this part is not to undertake a thorough analysis of the published algorithms, rather explain our approach to the large-scale generation using the given methods. For the details of the original algorithms, see the individual standards. Although NUMS, Brainpool and Curve25519 provide a method for generating group generators, we are currently not focusing on their analysis.

X9.62 - the standard. We focus on the generation method of the 1998 version [Com+98]. Its input is a 160-bit seed and a large prime p and the output is an elliptic curve in short Weierstrass form over the field $\mathbb{F}_p$, satisfying the following security conditions:

- "Near-primality": The curve order shall have a prime factor l of size at least $\min\{2^{160}, 4\sqrt{p}\}$. Furthermore, the cofactor shall be s-smooth, where s is a small integer (the standard proposes $s = 255$ as a guide).
- The embedding degree of the curve shall be greater than 20. The standard also specifies that to check this condition, we may simply verify that $p^e \neq 1 \pmod{l}$ for all $e \leq 20$.
- The trace t shall not be equal to 1.

Given a seed and a prime p, the standard computes a $\log(p)$-bit integer r using[11] the function (1). The next step is choosing $a, b \in \mathbb{F}_p$ such that $b^2 r = a^3$. This process is repeated until the curve satisfies the security conditions mentioned above.

$$H(\text{seed}) = \underbrace{\texttt{SHA-1}(\text{seed})}_{\text{discard}} || \underbrace{\texttt{SHA-1}(\text{seed}+1) || \ldots || \texttt{SHA-1}(\text{seed}+i)}_{\log(p)-\text{bit integer as output}}. \qquad (1)$$

X9.62 – our approach. For each of the five bit-lengths 128, 160, 192, 224 and 256 bits we have fixed the same prime as the standard (hence all curves of the same bit-length are defined over the same field). Since there is no guidance in the standard how to pick the seed for each iteration, we have taken the published seed for each bit-length and iteratively incremented its value by 1. To pick a and b, we have fixed $a = -3$ (as was done for most X9.62 curves for performance reasons [CC86]) and computed b accordingly, discarding the curve if $b^2 = -27/r$ does not have a solution for $b \in \mathbb{F}_p$. We also restricted the accepted cofactors to 1, 2, or 4 as this significantly accelerated the point counting. This choice also

[10] This took up to a week per standard on 40-core cluster of Intel Xeon Gold 5218.
[11] More precisely, H also changes the most significant bit of the output to 0.

conforms to with the fact that the standard curves all have cofactor 1 and the SECG standard (which overlaps with the X9.62 specifications) recommends the cofactor to be bounded by 4. The point counting – the main bottleneck of the computations – was done by an early-abort version of the SEA algorithm [Sch95]. For each of the five bit-lengths we have tried 5 million seeds, resulting in over 120 000 elliptic curves. Figure 3 captures a simplified overview of the algorithm.

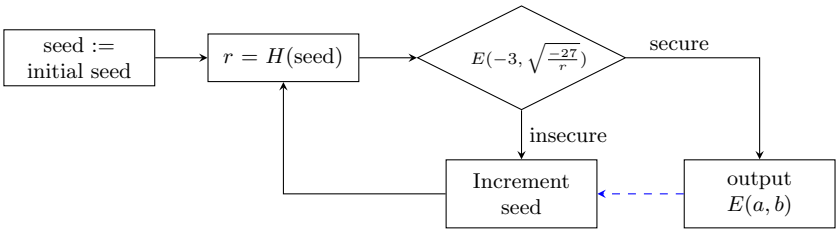

Fig. 3: X9.62 algorithm adjusted (indicated by the dashed line) for large-scale generation.

Brainpool – the standard. The Brainpool standard proposes algorithms for generating both the prime p and the curve over $\mathbb{F}_p$. Since in our simulations we have used the same finite fields as are in the recommended curve parameters we will skip the algorithm for prime generation.

Given a seed and a prime, the curve generation process outputs an elliptic curve in Weierstrass form that satisfies the following security conditions:

- The cofactor shall be 1, i.e. the group order n shall be prime.
- The embedding degree shall be greater than $(n-1)/100$.
- The trace t shall not be equal to 1. Technical requirements then state that $t > 1$.
- The class number of the endomorphism algebra of the curve should be larger than 10^7.

The algorithm itself follows similar idea as X9.62, but in more convoluted way, as can be seen in Figure 4. This time, the H function (1) is used to compute both a and b in pseudorandom way. Roughly speaking, a given seed is repeatedly incremented by 1 and mapped by H until an appropriate a is found and the resulting seed is used for finding b in a similar way. If the curve does not satisfy the condition, the seed is incremented and used again as the initial seed. See the standard for details of generation, GenA and GenB represent generation of a and b in Figure 4.

Brainpool – our approach. We have used the same approach as in X9.62 and used the published seed as initial seed for the whole generation, incrementing seed after each curve was found. Since generation of both a and b can in theory take an arbitrary number of attempts, the seed is incremented after each failed

attempt. We have again used 5 million seeds for each of the four bit-lengths (160, 192, 224, 256) Brainpool recommends. The number of generated curves in total is over 12 000. The drop in the proportion of generated curves compared to X9.62 standard is caused by stricter conditions.

There is currently no known efficient method that computes the class number; instead we checked that a related quantity – the CM discriminant – is greater than 2^{100}, following the SafeCurves recommendations [BL].

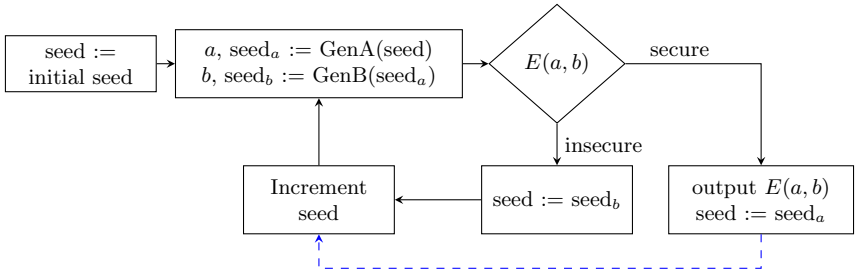

Fig. 4: Brainpool algorithm adjusted (indicated by the dashed line) for large-scale generation.

NUMS – the standard. The NUMS generation method, in accordance with its name, does not fall into the verifiably pseudorandom category. As Brainpool, NUMS proposes algorithms for generating both the prime field and the curve. The lowest recommended bit-length for prime fields by NUMS is 256. The method of prime generation works by starting with $c = 1$ and incrementing this value by 4 until $p = 2^s - c$ is a prime congruent to 3 mod 4.

Although NUMS proposes algorithms for generating both Weierstrass and Edwards curves, we have focused only on the Weierstrass curves. The curve is found by searching for $E(-3, b)$ satisfying the following security conditions by incrementing b, starting from $b = 1$:

- The curve order as well as the order of its twist shall be primes.
- The trace t shall not equal 0 or 1. This condition is further extended by requiring $t > 1$, supposedly for practical reasons.
- The embedding degree shall be greater than $(n-1)/100$ following the Brainpool standard.
- The CM discriminant shall be greater than 2^{100}, following the SafeCurves recommendations.

NUMS – our approach. We have used the same process, but this time iterating over 10 million values for b, for each of the four bit-lengths 160, 192, 224 and 256. Though the lowest recommended bit-length for prime fields by NUMS is 256, we have generated 160, 192 and 224-bit primes using the proposed method to study curves of lower bit-lengths. This process produced over 1 200 curves, implying that the twist condition is strongly restrictive.

Curve25519 – the standard. Curve25519 was created in a similar rigid manner as NUMS curves by minimizing one coefficient of the curve equation. The prime $2^{255} - 19$ for the prime field was chosen to be close to a power of 2 and with bit-length slightly below multiple of 32 for efficiency of the field arithmetic. Curve25519 is a Montgomery curve of the form $y^2 = x^3 + Ax^2 + x$ with the smallest $A > 2$ such that $A - 2$ is divisible by 4 and:

- The cofactor of the curve and its twist shall be 8 and 4 respectively.
- The trace t shall not equal 0 or 1.
- The embedding degree shall be greater than $(n - 1)/100$.
- The CM discriminant shall be greater than 2^{100}.

Curve25519 – our approach. Similarly to the NUMS approach we have iterated over A for bit-lengths 159, 191 and 255 where we have picked primes $2^{159} - 91$ and $2^{191} - 19$ for lower bitlengths. Iterating over 10 million values for A resulted in roughly 804 curves in total.

Random – our approach. Since all of the simulated standards contain certain unique properties (small b for NUMS, small A for Curve25519, bit overlaps in Brainpool, $rb^2 = -27$ for X9.62) and curves of the same bit-length share a base field, we have also generated curves for bit-lengths 128, 160, 192, 224, 256 using the method depicted in Figure 5 that aims to avoid such biases. We call such curves Random, as this term clearly describes our intention. However, for practical reasons, we made the process deterministic using a hash function. For each curve, we generated a prime for the base field by hashing (SHA-512) a seed and taking the smallest prime bigger than this hash when interpreted as an integer. Curve coefficients were chosen using the hash function as well, see fig. 5. We kept such a curve if it was not anomalous, had cofactor 1, 2, 4 or 8 and satisfied the SafeCurves criteria on embedding degree and CM discriminant. We tried 5 million seeds, resulting in over 250 000 Random curves.

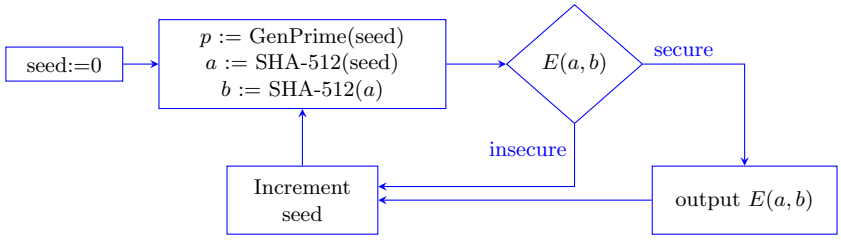

Fig. 5: Random simulation.

3.3 Outlier detection

Our framework offers options for graphical comparisons for described distinguishing strategies. However, manual inspection does not scale well, so we utilize

automated approaches to identify suspicious curves. Since we do not have (and cannot have) a labeling that could be used for the typical supervised approaches, we had to resort to unsupervised methods, namely outlier detection.

We built several datasets of simulated curves according to the selected distinguishing strategies. Each dataset of simulated curves consists only of curves of the same bit-length that were generated by the same method. Table 2 shows numbers of curves contained within each dataset. The rows correspond to the used generation method and the columns to the bit-length of the selected curves.

	256 bits	224 bits	192 bits	160 bits	128 bits
$\text{X9.62}_{\text{sim}}$	18 502	22 211	18 836	27 780	36 126
$\text{Brainpool}_{\text{sim}}$	1 677	2 361	2 640	3 184	0
NUMS_{sim}	83	109	191	325	0
$\text{Curve25519}_{\text{sim}}$	140	0	366	278	0
Random	18 636	21 226	24 805	29 639	37 311

Table 2: Numbers of curves in our datasets of simulated curves.

Additionally, in cases where trait results are (mostly) independent of the curve bit-length, we analyzed curves of all bit-lengths together. This approach allowed us to study even curves that did not match any of the generated bit-lengths.

We augmented each dataset with standard curves according to distinguishing strategies, i.e., Random curves with each category of standard curves, and the other three simulated categories only with standard curves of the same type. From these augmented datasets, we derived feature vectors consisting of all computed trait results, as well as some of their subsets. The features were scaled using a min-max scaler to fit within the $[0; 1]$ range. In case some results were not computed, they were replaced with -1, signifying that the given value took too long to compute (e.g., the factorization of large numbers). Finally, we ran the local outlier factor algorithm [Bre+00] to identify outliers within each augmented dataset. If the approach reported a standard curve as an outlier, we inspected such results closer.

4 Traits

DiSSECT currently contains 22 trait functions[12] – traits for brevity – designed to test a wide range of elliptic curve properties, including mathematical characteristics of curves, all SafeCurves ECDLP requirements, and properties connected to curve standards or implementations. In particular, the traits are not limited to properties connected to known attacks or vulnerabilities, and aim to find new interesting deviations.

[12] See `https://dissect.crocs.fi.muni.cz/` or Appendix A for a full list.

We have divided the traits into five categories:

Potential attacks. These traits test the classical properties of elliptic curves relevant to known attacks (the order and cofactor of the curve; the quadratic twist; the embedding degree). However, we also cover lesser-known and threatening attacks, for example, the factorization of values of the form $kn \pm 1$ as a generalization of [Che06]. To test scalar multiplication, essential to all ECC protocols, we inspect multiplicative orders of small values modulo the group order n. Another trait follows the idea from [Che02], with a possible connection of the discrete logarithm on ECC to factorization.

Complex multiplication. Although at this moment there is no serious attack utilizing any knowledge about the CM discriminant or class number, these values are the defining features of an elliptic curve. They determine the structure of the endomorphism ring, the torsion points, isogeny classes, etc. Multiple traits deal with the factorization of t, the size and the factorization of the Frobenius discriminant $D = t^2 - 4p$, as well as how D changes as we move the defining base field to its extensions. We also compute bounds on the class number using the Dirichlet class number formula [Dav80].

Torsion. We have designed several traits to directly analyze the torsion points of a given elliptic curve. One of the traits computes the degree of the extension $\mathbb{F}_{p^k}/\mathbb{F}_p$ over which the torsion subgroup is (partially) defined. Lower degrees might lead to computable pairings. Another trait approaches torsion from the direction of division polynomials and computes their factorization. We also consider the lift $E(\mathbb{F}_p) \to E(\mathbb{Q})$ and computes the size of the torsion subgroup over $\mathbb{Q}$ which might be relevant for lifting of ECDLP.

Isogenies. Both torsion and complex multiplication can be described using isogenies. Kernel polynomial of every isogeny is a factor of the division polynomial, and the special cases of isogenies, endomorphisms, form an order in an imaginary quadratic field of $\mathbb{Q}(\sqrt{D})$. One of isogeny traits computes the degree of the extension where some/all isogenies of a given degree are defined. Isogeny traits also analyze isogenies of ordinary curves and the corresponding isogeny volcanoes – their depth and the shape of so-called crater, and also the number of neighbors for a given vertex in the isogeny graph.

Standards and implementation. During our analysis of standard methods for generating curves, we have noticed unusual steps in the algorithms, so we designed traits to capture the resulting properties. The nature of the Brainpool standard (see Figure 4) causes overlaps in the binary representations of the curve coefficients in roughly half of the generated curves. The NUMS standard, as well as the Koblitz curves, use small curve coefficients by design. The X9.62 standard uses the relation $b^2r = a^3$ to determine the curve coefficients from r. The SECG standard recommends the curves `secp224k1` and `secp256k1` together with the group generators; the x-coordinate of half of both of these generators is the same small number.

Most attacks on the ECDLP are aiming at specific implementation vulnerabilities. To address this, we propose traits that target possible irregularities

in practical implementations. One trait follows the idea of [Wei+20], where the authors analyze the side-channel leakage caused by improper representation of large integers in memory. Based on [Bai+09], we designed a trait that analyzes the number of points with a low Hamming weight on a given curve.

4.1 Notable findings

GOST curves. The trait that analyzes the size of the coefficients in the Weierstrass form was motivated by the NUMS standard. However, our outlier detection also recognized two 256-bit GOST curves (`CryptoPro-A-ParamSet`, `CryptoPro-C-ParamSet`) in this trait, and a closer inspection in Figure 6 revealed that both curves have small b coefficients (166 and 32858). This contradicts Alekseev, Nikolaev, and Smyshlyaev [ANS18], who claim that all of the seven standardized GOST R curves were generated in the following way[13]:

1. Select p that allows fast arithmetic.
2. Compute r by hashing a random seed with the Streebog hash function.
3. For the generation of twisted Edwards curve $eu^2 + v^2 = 1 + du^2v^2$, put $e = 1, d = r$. For the generation of Weierstrass curve $y^2 = x^3 + ax + b$, put $a = -3$ and b equal to any value such that $rb^2 = a^3$.
4. Check the following security conditions:
   - $n \in (2^{254}, 2^{256}) \cup (2^{508}, 2^{512})$.
   - The embedding degree is at least 32 (resp. 132) if $n \in (2^{254}, 2^{256})$ (resp. if $n \in (2^{508}, 2^{512})$).
   - The curve is not anomalous.
   - The j-invariant is not 0 or 1728.

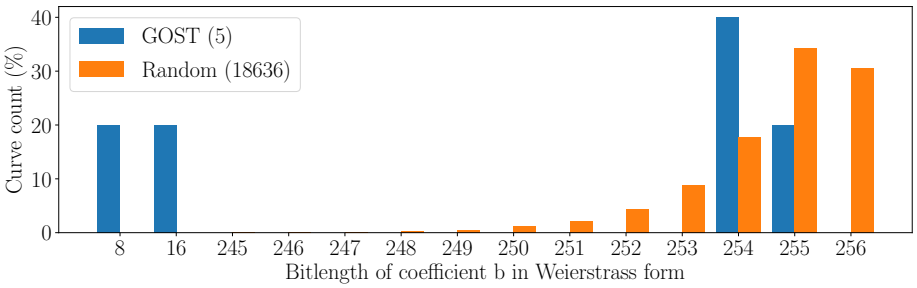

Fig. 6: Two GOST curves (`CryptoPro-A-ParamSet`, `CryptoPro-C-ParamSet`) exhibit particularly low bit-length of b parameter, even though Alekseev, Nikolaev, and Smyshlyaev [ANS18] claim that they were generated pseudorandomly.

[13] The authors support their claims by providing seeds for two of the seven curves. We find it problematic that the seeds were not previously made public.

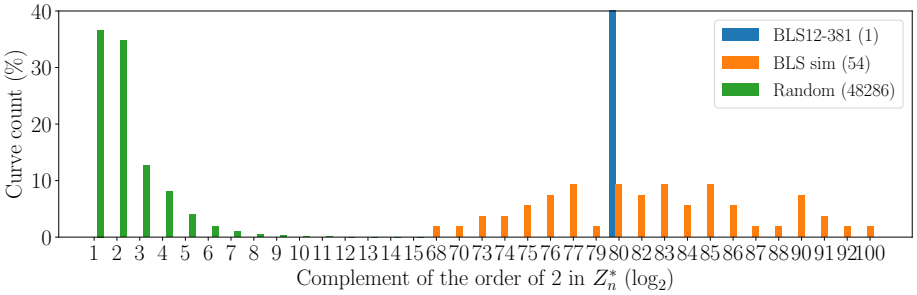

Fig. 7: BLS12-381 and BLS simulated curves exhibit unexpectedly large complement of the order of 2 in $\mathbb{Z}_n^*$ when compared to Random curves.

Thus the small size of b (which should be pseudorandom if r is) implies that it is very unlikely that they were generated with this claimed method. More precisely, since $b^2 = -27/r$ in $\mathbb{F}_p$ and p has 256 bits, then the probability for $b = 166$, resp. $b = 32858$ is 2^{-248}, resp. 2^{-240}, considering the bitlengths of the parameters. We hypothesise that the `CryptoPro-A-ParamSet` curve was generated by incrementing b from 1 until the GOST security conditions were satisfied. We have verified that $b = 166$ is the smallest such value with the added condition that the cofactor is 1 (otherwise, the smallest value is $b = 36$). The `CryptoPro-C-ParamSet` does not have this property, as its b coefficient 32858 is only the 80th smallest such value. However, there is an additional problem with the generator point $(0, \sqrt{32858})$. It was shown [Gou03], prior to the standardization of this curve, that there exist side-channel attacks utilizing such special points with $x = 0$. If we consider the existence of such point as another condition imposed on the curves, then 32858 is the 46th smallest value. We employed DiSSECT to distinguish this particular curve for from the rest of the 45 curves using the implemented traits in order to explain their choice, but without results.

Furthermore, the trait inspecting CM discriminant revealed that the third curve `CryptoPro-B-ParamSet` from [PLK06] has a CM discriminant of -619. Such a small value is extremely improbable, unless the curve was generated by the CM method [Brö06]. (The CM discriminant -915 of `gost256` is small as well, but this curve was used just as an example and there are no claims about its generation.)

The BLS12-381 Curve. The trait that measures $\phi(n)$ divided by the multiplicative order of 2 modulo n (low multiplicative orders translate to high values and vice versa) identified the BLS12-381 curve [Bow17] as an outlier in the set of Random curves. While some traits were expected to show statistical differences between pairing-friendly curves and Random curves, the cause was not clear for this particular trait. To further investigate this, we have adopted the generation method of the BLS12-381 curve to DiSSECT. Figure 7 shows the results of the trait on Random curves, the BLS12-381 curve and the BLS simulated curves.

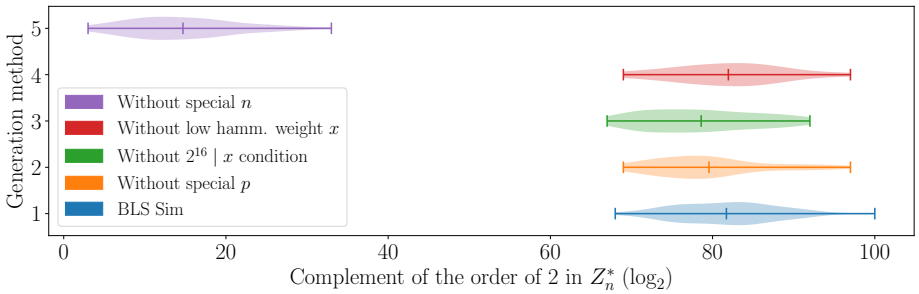

Fig. 8: Simulation of BLS curves while ignoring different properties have shown that the special construction of n caused the unexpected behavior.

Figure 7 shows that the unexpected detection of BLS12-381 as an outlier was not a coincidence. Rather it was caused by the generation method, and consequently the properties, of the BLS12-381 curve. To briefly summarize, the generation method works by finding an integer x satisfying:

- x has low hamming weight,
- x is divisible by 2^{16},
- $n = x^4 - x^2 + 1$ is a prime,
- $p = \frac{1}{3}n(x-1)^2 + x$ is a prime.

When such x is found, the CM method is used to construct a BLS curve[14] over $\mathbb{F}_p$ with cardinality $\frac{1}{3}n(x-1)^2$. We have identified exactly what conditions in the BLS generation method were behind the outlier detection of BLS12-381 curve by removing each condition and computing the trait for these modified curves. On one hand, Figure 8 shows that the conditions on x and special form of p have little or no impact on the results of the trait. On the other hand, we can see that the conditions that were the main cause of the unexpected results of BLS12-381 is the special form of the order $n = x^4 - x^2 + 1$. Indeed, in this case $\phi = x^2(x-1)(x+1)$ which means that small multiplicative orders are more likely than for general n. Although this does not seem to cause any vulnerabilities in the BLS12-381 curve, it reveals an undocumented property of the generation method.

The Bitcoin curve. A trait that inspects the x-coordinate of inverted generator scalar multiples, i.e., x-coordinates of points $k^{-1}G$, where $k \in \{1, \ldots, 8\}$, was inspired by Brengel et al. [BR18], who reported an unexpectedly low value for $k = 2$ on the Bitcoin curve `secp256k1`. Furthermore, Maxwell [Max15] pointed out that `secp224k1` yields exactly the same result. Pornin [Por19] guesses that this was caused by reusing the code for Koblitz curves with the same seeds, together with poor documentation.

[14] Iterating through 100 million values of x we have generated 54 BLS curves.

We analyzed the results of this trait in our visualization framework (Figure 9) and discovered that `secp256k1` and `secp224k1` are the only standard curves for which the x-coordinate of $k^{-1}G$ is significantly shorter than the full bit-length.

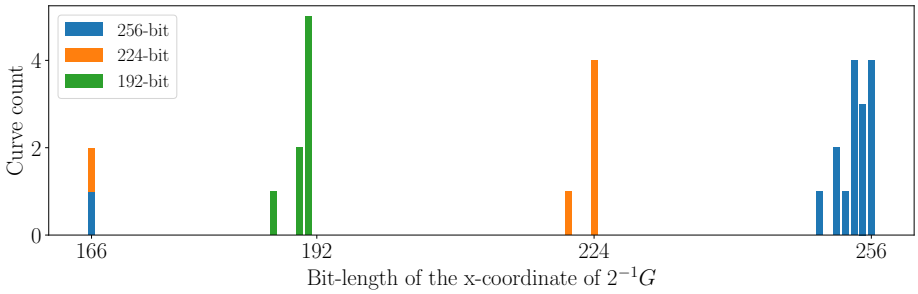

Fig. 9: Curves `secp256k1` and `secp224k1` exhibit significantly shorter x-coordinate of the point $2^{-1}G$ than expected. The other standard curves of given bit-lengths are also plotted.

Brainpool overlaps. The trait designed to detect bit-overlaps in the Weierstrass coefficients, revealed a structure in the Brainpool curves, already observed by Bernstein et al. [Ber+15].

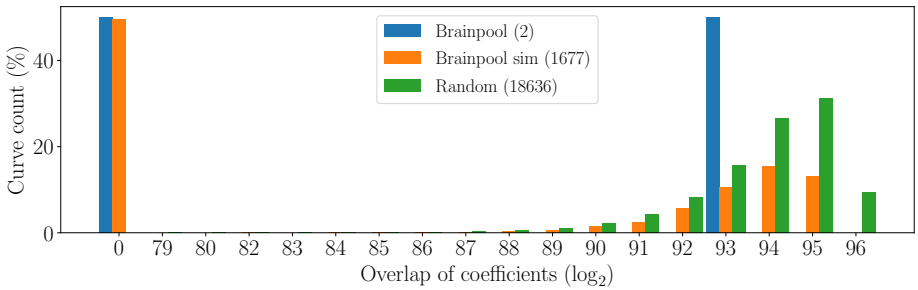

Fig. 10: Bit-overlap of 256-bit standard Brainpool, simulated Brainpool, and Random curves.

More precisely the trait compares coefficients a and b when stripped of 160 bits (SHA-1 output size) from the left and right respectively. Inspecting `brainpoolP256r1`, we can see the identical segments in a, b:

$a =$ `0x7d5a0975fc2c3057eef67530417affe7fb8055c126dc5c6ce94a4b44f330b5d9`

$b =$ `0x26dc5c6ce94a4b44f330b5d9bbd77cbf958416295cf7e1ce6bccdc18ff8c07b6`

Such overlaps occur during roughly half of the time during generation; e.g., for `brainpoolP{192,256,384}` but not for `brainpoolP{160,224,320}`. Figure 10 illustrates this effect for the Brainpool standard and simulated curves.

5 Our tool DiSSECT

DiSSECT is an open-source tool for generating elliptic curves according to our standard simulation methods, computing traits on elliptic curves, and analyzing data using automated approaches and an easy-to-use visualization environment. The tool does not aim to perform rigorous statistical tests given the limited sample size of standard curves. Its code contains implementations of all 22 traits described in Appendix A, and it can be easily extended. Adding a new trait requires only a short description of its function and writing a few lines of Sage code that computes the new trait result for a given curve.

The tool provides an approach for automated outlier detection to systematically identify deviations. A Jupyter notebook environment can be used to perform a more detailed analysis of computed trait results. Both of these approaches allow configuring the subset of analyzed curves, traits, and their parameters, and can access locally stored data as well as our publicly accessible database.

Our website[15] `https://dissect.crocs.fi.muni.cz/` contains detailed information about curves in our database and corresponding trait results, and trait descriptions with statistics. For a more complex data inspection, the analysis environment configured with access to our database can be launched directly from the website.

6 Conclusions

Our framework DiSSECT aspires to survey all standard elliptic curves and to assist in identification of potential problems by comparing them to simulated ones and visualizing the results. We built it as a foundation of elliptic curve cryptanalysis for the cryptographic community and hope that more cryptographers and mathematicians will join the project. DiSSECT's code is available at our repository[16].

Our tool revealed two surprising types of deviations. We realized that the generation process of three GOST curves described by Alekseev, Nikolaev, and Smyshlyaev [ANS18] is inconsistent with the sizes of the b coefficient in two cases and with the size of the CM discriminant in the third one. Properly documenting such properties is crucial for re-establishing trust in the standard curves and the whole ECC ecosystem. We cannot expect its users, developers, or policy-makers to notice even fairly obvious deviations (e.g., small Weierstrass coefficients), as they often do not access the parameters directly. **Follow up:** A discussion with the authors of [ANS18] revealed that the perceived contradiction was caused

[15] `https://dissect.crocs.fi.muni.cz/`
[16] `https://github.com/crocs-muni/DiSSECT`

by a different understanding of the word "standard". They do not consider the RFC document [PLK06] (which specifies the CryptoPro-X-ParamSet curves) as a standard and therefore did not mean to claim anything about the generating method of these curves.

We found an interesting, previously undescribed property of the BLS12-381 curve (related to smoothness) that is caused by its generation. Recent attacks [KB16] on special properties of pairing-friendly curves have proven that awareness of all properties caused by the pairing-friendly generation methods is crucial. In particular, our approach of isolation of individual properties in BLS generation has shown to have a lot of potential for future research of security of pairing-friendly curves. One further interesting improvement of DiSSECT would be an implementation of clustering algorithms that would help uncover biases systematically introduced by these properties. Another approach might be combining DiSSECT with statistical tools such as [Obr+15].

Selected parts of DiSSECT could also be used to quickly assess new individual curves. This might be useful for implementations following the idea of Miele and Lenstra [ML15], trading standard curves for ephemeral on-the-fly generated ones. Besides cryptographic applications, DiSSECT might also be useful to number theorists by providing them with empiric distributions of various traits. It is unrealistic to go through the whole space of trait results for different parameter choices and curve sets manually. Thus we employed an automated outlier detection method, which found all discrepancies we discovered manually. Still, there may be other outliers, and we believe it is an interesting open problem to statistically evaluate the results in a way that takes into account the inner structure of the data for a given trait.

Acknowledgements. This project has been made possible in part by a grant from the Cisco University Research Program Fund, an advised fund of Silicon Valley Community Foundation. V. Sedlacek, V. Matyas, M. Sys and A. Dufka were supported by Czech Science Foundation project GA20-03426S. V. Sedlacek and V. Suchanek were also supported by the Ph.D. Talent Scholarship - funded by the Brno City Municipality. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

References

[Age11] Agence Nationale de la Securite des Systemes d'Information. *Avis relatif aux paramètres de courbes elliptiques définis par l'Etat français.* 2011. URL: `https://www.legifrance.gouv.fr/download/pdf?id=QfYWtPSAJVtAB_c6Je5tAv00OY2r1-ad3LaVVmnStGvQ=`.

[ANS18] E. K. Alekseev, V. Nikolaev, and S. V. Smyshlyaev. "On the security properties of Russian standardized elliptic curves". In: *Mathematical questions of cryptography* 9.3 (2018), pp. 5–32.

[ANS05] ANSI. *Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*. ANSI X9.62. 2005.

[Ara+13] D. F. Aranha1, P. S. L. M. Barreto, G. C. C. F. Pereira, and J. E. Ricardini. *A note on high-security general-purpose elliptic curves*. Cryptology ePrint Archive, Report 2013/647. `https://eprint.iacr.org/2013/647`. 2013.

[Bai+15] T. Baigneres, C. Delerablée, M. Finiasz, L. Goubin, T. Lepoint, and M. Rivain. "Trap Me If You Can-Million Dollar Curve." In: *IACR Cryptol. ePrint Arch.* 2015 (2015), p. 1249.

[Bai+09] D. V. Bailey, L. Batina, D. J. Bernstein, P. Birkner, J. W. Bos, H.-C. Chen, C.-M. Cheng, G. Van Damme, G. de Meulenaer, L. J. D. Perez, et al. "Breaking ECC2K-130." In: *IACR Cryptol. ePrint Arch.* 2009 (2009), p. 541.

[BLS02] P. S. Barreto, B. Lynn, and M. Scott. "Constructing elliptic curves with prescribed embedding degrees". In: *International Conference on Security in Communication Networks*. Springer. 2002, pp. 257–267.

[Ber06] D. J. Bernstein. "Curve25519: new Diffie-Hellman speed records". In: *International Workshop on Public Key Cryptography*. Springer. 2006, pp. 207–228.

[Ber+15] D. J. Bernstein, T. Chou, C. Chuengsatiansup, A. Hülsing, E. Lambooij, T. Lange, R. Niederhagen, and C. Van Vredendaal. "How to Manipulate Curve Standards: A White Paper for the Black Hat http://bada55. cr. yp. to". In: *International Conference on Research in Security Standardisation*. Springer. 2015, pp. 109–139.

[BCL14] D. J. Bernstein, C. Chuengsatiansup, and T. Lange. "Curve41417: Karatsuba revisited". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2014, pp. 316–334.

[Ber+13] D. J. Bernstein, M. Hamburg, A. Krasnova, and T. Lange. "Elligator: Elliptic-curve points indistinguishable from uniform random strings". In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013, pp. 967–980.

[BLN16] D. J. Bernstein, T. Lange, and R. Niederhagen. "Dual EC: A standardized back door". In: *The New Codebreakers*. Springer, 2016, pp. 256–281.

[BL] D. J. Bernstein and T. Lange. *SafeCurves: choosing safe curves for elliptic-curve cryptography*. `https://safecurves.cr.yp.to/`. (Visited on 08/17/2021).

[BD00] E. Biham and O. Dunkelman. "Cryptanalysis of the A5/1 GSM stream cipher". In: *International Conference on Cryptology in India*. Springer. 2000, pp. 43–51.

[Bla+14] B. Black, J. Bos, C. Costello, P. Longa, and M. Naehrig. *Elliptic Curve Cryptography (ECC) Nothing Up My Sleeve (NUMS) Curves and Curve Generation*. Internet-Drafts are working documents of

the Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-black-numscurves-02. 2014.

[Bow17] S. Bowe. *BLS12-381: New zk-SNARK Elliptic Curve Construction*. https://electriccoin.co/blog/new-snark-curve/. 2017. (Visited on 09/16/2021).

[BGH19] S. Bowe, J. Grigg, and D. Hopwood. *Recursive Proof Composition without a Trusted Setup*. Cryptology ePrint Archive, Report 2019/1021. https://ia.cr/2019/1021. 2019.

[BR18] M. Brengel and C. Rossow. "Identifying Key Leakage of Bitcoin Users". In: *Research in Attacks, Intrusions, and Defenses*. Ed. by M. Bailey, T. Holz, M. Stamatogiannakis, and S. Ioannidis. Cham: Springer International Publishing, 2018, pp. 623–643.

[Bre+00] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. "LOF: identifying density-based local outliers". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 93–104.

[Brö06] R. Bröker. "Constructing elliptic curves of prescribed order". PhD thesis. Thomas Stieltjes Institute for Mathematics, 2006.

[Cer10] Certicom Research. *SEC 2: Recommended Elliptic Curve Domain Parameters, Version 2.0*. 2010. URL: https://secg.org/.

[Cer] Certicom Research. *Standards for Efficient Cryptography Group*. https://secg.org/. (Visited on 08/13/2021).

[Che+] S. Checkoway, J. Maskiewicz, C. Garman, J. Fried, S. Cohney, M. Green, N. Heninger, R. Weinmann, E. Rescorla, and H. Shacham. "A Systematic Analysis of the Juniper Dual EC Incident". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pp. 468–479.

[Che02] Q. Cheng. *A New Special-Purpose Factorization Algorithm. Citeseer*. 2002.

[Che06] J. H. Cheon. "Security Analysis of the Strong Diffie-Hellman Problem". In: *Advances in Cryptology - EUROCRYPT 2006*. Ed. by S. Vaudenay. Springer Berlin Heidelberg, 2006.

[CC86] D. V. Chudnovsky and G. V. Chudnovsky. "Sequences of numbers generated by addition in formal groups and new primality and factorization tests". In: *Advances in Applied Mathematics* 7.4 (1986), pp. 385–434.

[Col85] Collectif. "Nombres de classes des corps quadratiques imaginaires". In: *Séminaire Bourbaki: 1983/84, exposés 615-632*. Astérisque 121-122. Société mathématique de France, 1985. URL: http://www.numdam.org/item/SB_1983-1984__26__309_0/.

[Com+98] A. S. Committee et al. "American National Standard X9. 62-1998". In: *Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)* (1998).

[CLN15] C. Costello, P. Longa, and M. Naehrig. "A brief discussion on selecting new elliptic curves". In: *Microsoft Research. Microsoft* 8 (2015).

[Dav80] H. Davenport. *Multiplicative number theory*. 1980, pp. 43–53.

[DY15] L. Di and L. Yan. *Introduction to the Commercial Cryptography Scheme in China*. `https://icmconference.org/wp-content/uploads/C23Introduction-on-the-Commercial-Cryptography-Scheme-in-China-20151105.pdf`. 2015. (Visited on 08/23/2021).

[Brainpool] *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation*. Tech. rep. IETF RFC 5639, 2010.

[FR94] G. Frey and H.-G. Rück. "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves". In: *Mathematics of Computation* 62.206 (1994), pp. 865–874.

[GLV01] R. P. Gallant, R. J. Lambert, and S. A. Vanstone. "Faster point multiplication on elliptic curves with efficient endomorphisms". In: *Annual International Cryptology Conference (CRYPTO 2001)*. Springer. 2001, pp. 190–200. ISBN: 978-3-540-44647-7.

[Gou03] L. Goubin. "A refined power-analysis attack on elliptic curve cryptosystems". In: *International Workshop on Public Key Cryptography*. Springer. 2003, pp. 199–211.

[Hal13] T. C. Hales. "The NSA back door to NIST". In: *Notices of the AMS* 61.2 (2013), pp. 190–192.

[Ham15a] M. Hamburg. *A note on high-security general-purpose elliptic curves*. Cryptology ePrint Archive, Report 2015/625. `https://eprint.iacr.org/2015/625.pdf`. 2015.

[Ham15b] M. Hamburg. *Ed448-Goldilocks, a new elliptic curve*. IACR Cryptology ePrint Archive, Report 2015/625. 2015.

[Hop20] D. Hopwood. *The pasta curves*. `https://electriccoin.co/blog/the-pasta-curves-for-halo-2-and-beyond/`. 2020. (Visited on 08/27/2021).

[ISO17] ISO/IEC 15946. *Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 5: Elliptic curve generation*. 2017.

[Jou11] Journal officiel de la republique francaise. *Avis relatif aux paramètres de courbes elliptiques définis par l'Etat français*. 2011. URL: `https://www.legifrance.gouv.fr/download/pdf?id=QfYWtPSAJVtAB_c6Je5tAv00OY2r1ad3LaVVmnStGvQ=`.

[KB16] T. Kim and R. Barbulescu. "Extended tower number field sieve: A new complexity for the medium prime case". In: *Annual International Cryptology Conference*. Springer. 2016, pp. 543–571.

[KM16] N. Koblitz and A. Menezes. "A riddle wrapped in an enigma". In: *IEEE Security & Privacy* 14.6 (2016), pp. 34–42.

[Loc+14] M. Lochter, J. Merkle, J.-M. Schmidt, and T. Schutze. *Requirements for Standard Elliptic Curves*. IACR Cryptology ePrint Archive, Report 2014/832. 2014.

[Max15] G. Maxwell. *The most repeated R value on the blockchain*. `https://bitcointalk.org/index.php?topic=1118704.0`. 2015. (Visited on 09/09/2021).

[MOV93] A. J. Menezes, T. Okamoto, and S. A. Vanstone. "Reducing elliptic curve logarithms to logarithms in a finite field". In: *IEEE Transactions on information Theory* 39.5 (1993), pp. 1639–1646.

[ML15] A. Miele and A. K. Lenstra. "Efficient ephemeral elliptic curve cryptographic keys". In: *International Conference on Information Security*. Springer. 2015, pp. 524–547.

[MIR18] MIRACL UK Ltd. *Multiprecision Integer and Rational Arithmetic Cryptographic Library*. `https://github.com/miracl/MIRACL`. 2018.

[MNT01] A. Miyaji, M. Nakabayashi, and S. Takano. "New Explicit Conditions of Elliptic Curve Traces for FR-Reduction". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 84 (2001), pp. 1234–1243.

[Nak08] S. Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: *Decentralized Business Review* (2008), p. 21260.

[Nat00] National Institute of Standards and Technology. *FIPS Publication 186-2*. 2000. URL: `https://csrc.nist.gov/publications/detail/fips/186/2/archive/2000-01-27`.

[Obr+15] L. Obrátil, D. Klinec, P. Švenda, and M. Ukrop. *Randomness Testing Toolkit*. `https://rtt.ics.muni.cz`. 2015.

[Orm98] H. Orman. *The OAKLEY Key Determination Protocol*. RFC 2412. Nov. 1998. URL: `https://rfc-editor.org/rfc/rfc2412.txt`.

[Per+11] G. C. Pereira, M. A. Simplício Jr, M. Naehrig, and P. S. Barreto. "A family of implementation-friendly BN elliptic curves". In: *Journal of Systems and Software* 84.8 (2011), pp. 1319–1326.

[PH78] S. Pohlig and M. Hellman. "An Improved Algorithm for Computing Logarithms over $GF(p)$ and Its Cryptographic Significance". In: *IEEE Transactions on Information Theory* (1978), pp. 106–110.

[PLK06] V. Popov, S. Leontiev, and I. Kurepkin. *Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms*. 2006. URL: `https://rfc-editor.org/rfc/rfc4357.txt`.

[Por19] T. Pornin. `https://crypto.stackexchange.com/questions/60420/what-does-the-special-form-of-the-base-point-of-secp256k1-allow`. 2019. (Visited on 09/16/2021).

[SA+98] T. Satoh, K. Araki, et al. "Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves". In: *Rikkyo Daigaku sugaku zasshi* 47.1 (1998), pp. 81–92.

[Sch13] B. Schneier. *The NSA Is Breaking Most Encryption on the Internet*. `https://www.schneier.com/blog/archives/2013/09/the_nsa_is_brea.html`. 2013. (Visited on 08/13/2021).

[Sch95] R. Schoof. "Counting points on elliptic curves over finite fields". In: *Journal de Théorie des Nombres de Bordeaux* (1995), pp. 219–254.

[Sco99] M. Scott. *Re: NIST annouces set of Elliptic Curves*. https://web.archive.org/web/20160313065951/https://groups.google.com/forum/message/raw?msg=sci.crypt/mFMukSsORmI/FpbHDQ6hM_MJ. 1999. (Visited on 08/13/2021).

[Sco15] M. Scott. *A new curve*. https://moderncrypto.org/mail-archive/curves/2015/000449.html. 2015. (Visited on 09/01/2021).

[Sed22] V. Sedláček. "On cryptographic weaknesses related to elliptic curves". PhD thesis. Masaryk University, 2022.

[Sem96] I. Semaev. "On computing logarithms on elliptic curves". In: *Discrete Mathematics and Applications* 6.1 (1996), pp. 69–76.

[Sem98] I. Semaev. "Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p". In: *Mathematics of computation* 67.221 (1998), pp. 353–356.

[SSL14] S. Shen, S. Shen, and X. Lee. *SM2 Digital Signature Algorithm*. Internet-Draft. Work in Progress. 2014. URL: https://datatracker.ietf.org/doc/html/draft-shen-sm2-ecdsa-02.

[Sma99] N. P. Smart. "The discrete logarithm problem on elliptic curves of trace one". In: *Journal of cryptology* 12.3 (1999), pp. 193–196.

[SF16] E. S. Smyshlyaeva and V. Fotieva. *Information technology. Cryptographic data security. Parameters of elliptic curves for cryptographic algorithms and protocols*. Federal Agency on Technical Regulating and Metrology. 2016.

[Sol11] J. A. Solinas. "Generalized Mersenne Prime". In: *Encyclopedia of Cryptography and Security*. Ed. by H. C. A. van Tilborg and S. Jajodia. Springer US, 2011.

[Tor94] C. E. Torkelson. "The Clipper Chip: How Key Escrow Threatens to Undermine the Fourth Amendment". In: *Seton Hall L. Rev.* 25 (1994), p. 1142.

[Val+18] L. Valenta, N. Sullivan, A. Sanso, and N. Heninger. *In search of CurveSwap: Measuring elliptic curve implementations in the wild*. Cryptology ePrint Archive, Report 2018/298. https://ia.cr/2018/298. 2018.

[Wei+20] S. Weiser, D. Schrammel, L. Bodner, and R. Spreitzer. "Big Numbers - Big Troubles: Systematically Analyzing Nonce Leakage in (EC)DSA Implementations". In: *USENIX Security Symposium*. 2020.

[Wir00] Wireless Application Protocol Forum. *Wireless Application Protocol Wireless Transport Layer Security Specification*. https://web.archive.org/web/20170829023257/https://www.wapforum.org/tech/documents/WAP-199-WTLS-20000218-a.pdf. 2000.

A List of traits

The following is a list of all traits used for curve analysis. Each trait takes as an input an ordinary elliptic curve $E/\mathbb{F}_p : y^2 = x^3 + ax + b$ of order n.

trait name	**input**	**output**
`cofactor`	int r	Tuple (n_1, n_2) such that the group $E(\mathbb{F}_{p^r})$ is isomorphic to $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ and $n_1 \mid n_2$ ($n_1 = 1$ for cyclic groups).
`discriminant`		The factorization of $D = t^2 - 4p = v^2 d_K$, where d_K is the discriminant of the endomorphism algebra of E.
`twist order`	int r	The factorization of the cardinality of the quadratic twist of $E(\mathbb{F}_{p^r})$.
`kn fact.`	int k	The factorizations of $kn + 1$, $kn - 1$
`torsion extension`	prime l	$k_1, k_2, k_2/k_1$, where k_1, k_2 are the smallest integers satisfying $E[l] \cap E(\mathbb{F}_{p^{k_1}}) \neq \emptyset$ and $E[l] \subseteq E(\mathbb{F}_{p^{k_2}})$.
`conductor`	int r	The factorization of D_r/D_1, where $D_r = t_r^2 - 4p^r$ and t_r is the trace of Frobenius of $E/\mathbb{F}_{p^r}$.
`embedding`		The ratio $\phi(r)/e$, where $r \mid n$ is the order of the prime order subgroup and e is the multiplicative order of $q \pmod r$.
`class number`		Upper bound [Dav80] and lower bound [Col85] on the class number of the endomorphism algebra of E.
`small prime order`	prime l	The ratio $\phi(n)/m$ where m is the multiplicative order of $l \pmod n$.
`division pol.`	prime l	The factorization of the l-th division polynomial.
`volcano`	prime l	The depth and the degree of the l-volcano.
`isogeny extension`	prime l	$i_1, i_2, i_2/i_1$ where i_1, i_2 are the smallest integers such that there exists a $\mathbb{F}_{p^{i_1}}$-rational l-isogeny and $l+1$ $\mathbb{F}_{p^{i_2}}$-rational l-isogenies from E.
`trace fact.`	int r	A factorization of the trace of Frobenius of $E/\mathbb{F}_{p^r}$.
`isogeny neighbors`	prime l	A number of roots of $\Phi_l(j(E), x)$ where Φ_l is the l-th modular polynomial.
`q torsion`		Torsion order of $E'(\mathbb{Q})$ where E' is is given by the same equation $y^2 = x^3 + ax + b$.
`hamming x`	int k	A number of points on E with the Hamming weight of the x-coordinate equal to k.
`square 4p-1`		The factorization of square-free parts of $4p - 1$ and $4n - 1$ where n is the order of the generator point of E
`pow distance`		The distances of n to the nearest power of 2 and multiples of 32 and 64.
`multiples x`	int k	The x-coordinate of $\frac{1}{k}G$.
`x962 inv.`		The value $r = \frac{a^3}{b^2}$.
`brainpool overlap`		$a_{160} - b_{-160}$ where a_{160} are the $s - 160$ rightmost bits of a and b_{-160} are the $s - 160$ leftmost bits of b.
`weierstrass`		The parameters a, b.