Chaotic Key Generation using Elliptic Curves

Daniel Larsson

Abstract

In this short note we propose a method, generalising [3], for key generation using the group structure of an elliptic curve over a finite field, in conjunction with a chaotic map.

1 Introduction

In [3], Reyad and Kotulski introduced (and analysed statistically) a method for constructing pseudo-random numbers by using chaotic maps and the addition law on elliptic curves. A slight generalisation of this method was used in [2] to construct keys for use in biometric encryption. In this paper we significantly generalise the Reyad–Kotulski method by using n-dimensional chaotic maps and more points on the elliptic curve. As a consequence we can construct several keys in one go.

It is well worth remarking from the start that due to space limitations we do not present any numerical examples as even the simplest, non-trivial, such would invariably fill out significant portion (if not more) of the allotted space.

2 Discrete dynamical systems and chaotic maps

Let $\mathsf{Y} \subseteq \mathbb{R}^n$ and let $\Xi : \mathsf{Y} \to \mathsf{Y}$ be a map. A *discrete n-dimensional dynamical system on* Y is the set of *orbits*

$$\mathsf{orb}(\boldsymbol{x}_0) := \big\{(\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_n, \dots) \in (\mathbb{R}^n)^\infty \;\big|\; \boldsymbol{x}_{n+1} := \Xi(\boldsymbol{x}_n)\big\},$$

for all possible, initial vectors, the *seeds*, $\boldsymbol{x}_0$ in Y (whence all $\boldsymbol{x}_n \in \mathsf{Y}$). We denote the system simply by (Ξ, Y).

We will give a (very) heuristic definition of chaos. A *chaotic* (discrete) dynamical system is a system that is very sensitive to changes in the input seeds, in the sense that, for $\epsilon > 0$, the orbits $\mathsf{orb}(\boldsymbol{x}_0)$ and $\mathsf{orb}(\boldsymbol{x}_0 + \epsilon)$ diverge. This can be encapsulated in the catch phrase that chaotic systems have *Sensitive Dependence on Initial Conditions* (SDIC). Chaotic systems are (almost) always non-linear, i.e., Ξ is a non-linear map.

Example 1. Arguably, the simplest example of a chaotic dynamical system is the *logistic map* which is the map $\Xi(x) = \lambda x(1-x)$ on $\mathsf{Y} := [0,1]$ with $\mathsf{Y}_0 := [0,1/2)$, $\mathsf{Y}_1 := [1/2,1]$, and where $\lambda > 3.6$. There are several other one-dimensional chaotic systems, such as the *dyadic transformation*, that exhibit promising behaviour in the present context. A two-dimensional example, is the *Tinkerbell map*, given by

$$\Xi : \mathsf{Y} \to \mathsf{Y}, \qquad \begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} x_n^2 - y_n^2 + ax_n + by_n \\ 2x_ny_n + cx_n + dy_n \end{pmatrix}, \qquad \mathsf{Y} := [-2,1] \times [-2,1],$$

for suitable values on $a, b, c, d \in \mathbb{R}$. However, significant care needs to be taken if the Tinkerbell map is used in Algorithm 1 (see the remarks below).

Now, let $\mathsf{Y} \subset \mathbb{R}^n$ be a closed subset and let $\Xi : \mathsf{Y} \to \mathsf{Y}$ be a discrete dynamical system. We split $\mathsf{Y} := \mathsf{Y}_0 \cup \mathsf{Y}_1$, where $\mathsf{Y}_0 \cap \mathsf{Y}_1 = \emptyset$. Let Σ be a set of vectors

$$\Sigma := \big\{ \boldsymbol{\sigma}_i \in \mathsf{Y} \mid 1 \le i \le d \big\}, \quad d \ge 1.$$

This set will be the set of seeds for the dynamical system. We must require that

$$\mu(\Xi(\Sigma) \cap \mathsf{Y}_0) = \mu(\Xi(\Sigma) \cap \mathsf{Y}_1),$$

where μ is some measure on $\mathbb{R}^n$ and the equality is taken modulo some set of measure zero. Loosely speaking this requirement means that the orbits are "spread out evenly" over Y_0 and Y_1. In other words, every orbit passes through Y_0 and Y_1 equally often.

However, this is not sufficient: there must also be adequate "mixing" between Y_0 and Y_1, to give a suitable level of pseudo-randomness. This mixing property is difficult to define rigourosly and so we will simply say that when, for instance $x_n \in \mathsf{Y}_0$, the probability that $x_{n+1} \in \mathsf{Y}_1$ is generically not less than 1/2.

Not all chaotic systems satisfy this mixing property "on their own" (in other words, there is no natural splitting of Y into two disjoint subsets) and so a deeper, more technical, analysis of the system is needed to use it safely in the present context. As an example, the logistic map has sufficient mixing to ensure "randomness", while the Tinkerbell map does not "on its own". To use this, one needs to know more mathematical details on the Tinkerbell system in order to choose the correct Y_0 and Y_1, as well as the parameters a, b, c, d and suitable seeds. On the other hand, the so-called *Baker's map* seems to be a map that exhibits the necessary mixing "on its own".

Algorithm 1 now constructs a set of binary numbers. Notice that we for convenience restrict ourselves to even lengths of the orbits in the dynamical system. This is obviously not a limitation in practice. Let us re-iterate that, as stated, this algorithm does not ensure that the output is "random" enough. This failure of "randomness" can happen, for instance, if the system has a so-called "strange attractor". In such a case a more thorough analysis is needed.

3 Elliptic curves over finite fields

Let $\mathbb{F}_q$ denote the field of q elements (see, for instance, [1]). We take a pedestrian view on elliptic curves. Details can be found in, e.g., [1]. For a more in depth exposition than [1], we recommend the canonical reference [4].

Algorithm 1: Chaotic generation of binary numbers

```
Input: Chaotic dynamical system $(\Xi, \mathsf{Y})$, with $\mathsf{Y} \subset \mathbb{R}^n$ as above; Set of
       seeds $\Sigma$; A set of *even* numbers $\boldsymbol{\eta} := \{\eta_i \in \mathbb{Z}_{\gg 1} \mid 1 \leq i \leq d\}$
Output: Binary numbers $\mathbf{b} := \{\mathsf{b}_i \mid 1 \leq i \leq d\}$
1  $i := 1$
2  while $i \leq d$ do
3      $\boldsymbol{x} := \boldsymbol{\sigma}_i$
4      for $1 \leq j \leq \eta_i$ do
5          if $\boldsymbol{x} \in \mathsf{Y}_0$ then
6              $b_j := 0$
7          else
8              $b_j := 1$
9          $\boldsymbol{x} := \Xi(\boldsymbol{x})$
10     $\mathsf{b}_i := b_1 b_2 \cdots b_{\eta_i}$
11     $\mathsf{b}_i \to \mathbf{b}$
12     $i := i + 1$
13 $\mathbf{b} := \{\mathsf{b}_i \mid 1 \leq i \leq d\}$
```

Definition 3.1. Let $q = p^n$, with $p \geq 5$ a prime. An *elliptic curve*, $\mathscr{E}$ over $\mathbb{F}_q$, is a plane curve in $\mathbb{F}_q^2$ defined by the equation

$$y^2 = x^3 + \alpha x + \beta, \quad \text{with } \alpha, \beta \in \mathbb{F}_q, \tag{1}$$

and such that $4\alpha^3 + 27\beta^2 \neq 0$, together with a distinguished (abstract) element $\infty \notin \mathbb{F}_q^2$, the *point at infinity*. The set of solutions is denoted $\mathscr{E}(\mathbb{F}_q)$.

The set $\mathscr{E}(\mathbb{F}_q)$ is an abelian group under addition, which we denote $\underset{\mathscr{E}}{\oplus}$. For the actual formulas defining the group structure we point the reader to [4]. The element $\infty \in \mathscr{E}$ is the unit element for the group structure on $\mathscr{E}$.

4 Key generation

We start by defining a few functions that we will need. Let $\mathsf{b} = b_1 b_2 \ldots b_n$ be a binary number, where n is even. Put

$$\mathbf{s}^{\mathrm{L}}(\mathsf{b}) := b_1 b_2 \cdots b_{\frac{n}{2}}, \quad \text{and} \quad \mathbf{s}^{\mathrm{R}}(\mathsf{b}) := b_{\frac{n}{2}+1} b_{\frac{n}{2}+2} \cdots b_n.$$

In order to not inflict more notation on the reader, when $a \in \mathbb{F}_q$ and we write $\mathbf{s}^{\mathrm{L}}(a)$ or $\mathbf{s}^{\mathrm{R}}(a)$, we implicitly mean that a is given in binary form. Hence, implicit in this short-hand is a choice of binary function $B : \mathbb{F}_q \to \{0,1\}^{\times w}$, for some $w \in \mathbb{N}$ big enough, and so we write $\mathbf{s}^{\mathrm{L}}(a)$ instead of $\mathbf{s}^{\mathrm{L}}(B(a))$, and similarly with $\mathbf{s}^{\mathrm{R}}$.

If p is a (binary) point, we use the notation p_x and p_y to denote the x- and y-coordinates. Finally, if b is a binary number, we denote by $\mathsf{b}|_i$ the i-th bit in b.

Suppose $u \in \mathbb{F}_q$, and we want u to be the x-coordinate of a point on $\mathscr{E}$. For this to be possible we need to be able to solve the quadratic equation

$$y^2 = u^3 + \alpha u + \beta, \tag{2}$$

in $\mathbb{F}_q$. If the chosen u makes the equation non-solvable, we simply choose a new u until we get a u such that the equation (2) *is* solvable.

4.1 Algorithm 1: a single curve

We assume from now on that we have chosen a set $\{u_k \in \mathbb{F}_q \mid 1 \le k \le d\}$ such they are all x-coordinates of points in $\mathscr{E}(\mathbb{F}_q)$. We don't assume all the u_k are distinct, but we do assume that $\mathsf{p}_k, \mathsf{q}_k \neq \infty$ for every k.

Recall that d is the number of seeds into the chaotic machine. Now, let

$$\Phi := \Big\{ \phi_k : \mathbb{Z}/\eta_k \to \mathbb{Z}/\eta_k \mid 1 \le k \le d \Big\}$$

be a set of functions (for instance, elements in the symmetric group $\mathbb{S}_{\eta_k}$). In addition, we fix two sets of *ordered* points

$$\mathsf{P} := \Big\{ \mathsf{p}_k \in \mathscr{E}(\mathbb{F}_q) \mid 1 \le k \le d \Big\}$$

and

$$\mathsf{Q} := \Big\{ \mathsf{q}_k \in \mathscr{E}(\mathbb{F}_q) \mid 1 \le k \le d,\ (\mathsf{q}_k)_x = u_k \Big\}$$

such that $\mathsf{p}_i \neq \mathsf{q}_j$ for all $1 \le i, j \le d$. The points in P and Q need not be distinct. We view the points in Q as "base points" for the key generator.

Strictly speaking, we don't need to start with the u_k to construct the q_k, but as we will see this will allow us to better explain how to use the algorithm below to construct hashes.

Recall that b_k is a binary number generated with Algorithm 1. We now construct the following sequence

$$\Big\{ \mathsf{z}_j^k \in \mathscr{E}(\mathbb{F}_q) \mid 1 \le j \le \eta_k,\ 1 \le k \le d \Big\} \tag{3}$$

of points in $\mathscr{E}(\mathbb{F}_q)$, where the z_j^k are given by

$$\mathsf{z}_j^k := \phi_k(j)(1 + \mathsf{b}_k|_j)\mathsf{p}_k \underset{\mathscr{E}}{\oplus} \mathsf{q}_k, \quad \text{for} \quad 1 \le j \le \eta_k. \tag{4}$$

We will apply the operators $\mathbf{s}^{\mathrm{L}}$ and $\mathbf{s}^{\mathrm{R}}$ to these points

It is possible that $\mathsf{z}_j^k = \infty$ for some k and j. This can be resolved by, for instance, setting $\mathbf{s}^{\mathrm{L}}((\mathsf{z}_j^k)_x) := 10$ and $\mathbf{s}^{\mathrm{R}}((\mathsf{z}_j^k)_y) := 01$ or something to that effect.

If a and b are two (binary) numbers, $a \sqcup b$ denotes the (binary) number obtained by concatenation of a and b (hence, in general $a \sqcup b \neq b \sqcup a$). The key generation algorithm is now given as Algorithm 2.

The following is important enough to warrant its own remark.

Remark 1. Observe that the algorithm produces d keys. XOR:ing these keys to produce a single key, might counteract any defiency in the mixing property in the underlying dynamical system.

Remark 2. We feel that necessity calls for us to make the following remarks.

(0) **Comment on how this is a generalisation of the Reyad–Kotulski case.**

Algorithm 2: Key generation, one curve

```
Input: Set of binary numbers $\mathbf{b}$; Set of functions $\Phi$; Two ordered sets
       of points $\mathsf{P}, \mathsf{Q} \subset \mathscr{E}(\mathbb{F}_q)$
Output: Binary numbers $\mathbf{K} := \{\mathsf{K}_k \mid 1 \leq k \leq d\}$
1 for $1 \leq k \leq d$ do
2     Compute the points $\{\mathsf{z}_j^k \mid 1 \leq j \leq \eta_k\}$ from (4)
3     $\mathsf{K}_k := \mathbf{s}^{\mathrm{L}}((\mathsf{z}_1^k)_x) \bigsqcup \mathbf{s}^{\mathrm{R}}((\mathsf{z}_1^k)_y)$
4     for $2 \leq j \leq \eta_k$ do
5         if $j$ odd then
6             $\mathsf{K}_k := \mathsf{K}_k \bigsqcup \mathbf{s}^{\mathrm{L}}((\mathsf{z}_j^k)_x) \bigsqcup \mathbf{s}^{\mathrm{R}}((\mathsf{z}_j^k)_y)$
7         else
8             $\mathsf{K}_k := \mathsf{K}_k \bigsqcup \mathbf{s}^{\mathrm{L}}((\mathsf{z}_j^k)_y) \bigsqcup \mathbf{s}^{\mathrm{R}}((\mathsf{z}_j^k)_x)$
9 $\mathbf{K} := \{\mathsf{K}_k \mid 1 \leq k \leq d\}$
```

(1) The prime p should be "big". How "big", you ask? In [3] Reyad and Kotulski use primes that have ≈ 4 digits (in base-10). What is optimal with respect to computational cost and security in the present context is unclear and needs to be investigated, but intuitively it seems reasonable to expect the need for a bigger p.

(2) The order of points in P and Q is important. Any permutation of the points will in general (always?) give different keys, unless there is only one point in P and Q, of course. So, if $\tau \in \mathbb{S}_d$ is any permutation of d elements, then taking $\tau(\mathsf{P})$ and/or $\tau(\mathsf{Q})$ will give a different set of keys, using the same points on the curve.

(3) Clearly, there are the, not so small, issues with implementation, giving numerical examples and security analysis. As remarked on before, this is beyond the intended scope of this short note. However, see [2] for an example where the above strategy was used in the simplest case of the one-dimensional logistic map and where $k = 1$, in the context of construction of keys for biometric encryption.

(4) More freedom concerning points on the elliptic curves can be had by allowing points with coordinates in extension fields $\mathbb{F}_{q^a}$ of $\mathbb{F}_q$.

4.2 Application: constructing hashes

Is indicated above, Algorithm 2 can also serve as a basis for construction hashes. Indeed, let `msg` be a "message", represented as an element in $\mathbb{F}_p$, that we wish to hash. Assume to start with that `msg` is the x-coordinate of a point $\mathtt{q}$ on the elliptic curve $\mathscr{E}$. In order to be consistent with the notation above we put $u :=$ `msg`.

By the grace of Algorithm 1 we assume that we are given a chaotically generated binary number b. Pick a point $\mathtt{p} \in \mathscr{E}(\mathbb{F}_q)$ different from $\mathtt{q}$ and construct the sequence of points $\mathtt{z}_j$. Observe that we take $k = 1$ for now; in other words, we only use one seed. Then using Algorithm 2 we can construct the number K, which will be our hash value.

However, hash values associated with a fixed hashing algorithm all have the same length so we need to adjust for this. This can be achieved by choosing the length of the orbit (of the seed) to be fixed (and an even number).

Now, taking $k \geq 2$, we can do three things:

(1) construct k hashes in one go, or

(2) construct k hashes in one go, and then XOR the results, or

(3) hash several messages at once.

Obviously, a thorough analysis of the above construction needs to be done to evaluate the requirement for an algorithm to be a safe hash function.

4.3 Algorithm 3: multiple curves

Instead of using one curve with multiple seeds, parametrised by the index k, we can use k to parametrise pairs $(\mathscr{E}_k, \sigma_k)$, where the $\mathscr{E}_k$ are elliptic curves (not necessarily distinct) and $\sigma_k \in \Sigma$ a seed for the chaotic system. We could use several seeds for each curve, but for simplicity (of notation primarily) we associate one curve and one seed in the following.

Let

$$\mathsf{\Phi} := \Big\{ \phi_k : \mathbb{Z}/\eta_k \to \mathbb{Z}/\eta_k \mid 1 \leq k \leq d \Big\}$$

be a set of functions and $\mathbf{b} = \{\mathsf{b}_k\}$ a set of η_k-bit binary numbers, as before. Now, take a family of elliptic curves (not necessarily distinct) parameterised by $1 \leq k \leq d$,

$$\boldsymbol{\mathscr{E}} := \Big\{ \mathscr{E}_k : \; y^2 = x^3 + \alpha_k x + \beta_k \mid 1 \leq k \leq d, \;\; \alpha_k, \beta_k \in \mathbb{F}_q \Big\},$$

and elements $u_k \in \mathbb{F}_q$. We also allow ourselves two sets of points

$$\mathsf{P}_{\boldsymbol{\mathscr{E}}} := \Big\{ \mathsf{p}_k \in \mathscr{E}_k(\mathbb{F}_q) \mid 1 \leq k \leq d \Big\}$$

and

$$\mathsf{Q}_{\boldsymbol{\mathscr{E}}} := \Big\{ \mathsf{q}_k \in \mathscr{E}_k(\mathbb{F}_q) \mid 1 \leq k \leq d, \; (\mathsf{q}_k)_x = u_k \Big\}$$

such that $\mathsf{p}_k \neq \mathsf{q}_k$ for all $1 \leq k \leq d$.

Modifying the construction in (3), we do the following: construct the sequence of points z^k in $\mathscr{E}_k(\mathbb{F}_q)$ by

$$\mathsf{z}^k := \big(\mathsf{z}^k_j\big)_{j=1}^{\eta_k} \in \mathscr{E}_k(\mathbb{F}_q)^{\times \eta_k} := \prod_{j=1}^{\eta_k} \mathscr{E}_k(\mathbb{F}_q), \tag{5}$$

with

$$\mathsf{z}^k_j := \phi_k(j)(1 + \mathsf{b}_k|_j)\mathsf{p}_k \underset{\mathscr{E}_k}{\oplus} \mathsf{q}_k \in \mathscr{E}_k(\mathbb{F}_q)$$

as before (equation (4)). This gives us the d-tuple

$$\underline{\mathsf{z}} := \big(\mathsf{z}^1, \mathsf{z}^2, \ldots, \mathsf{z}^k\big) \in \prod_{k=1}^{d} \mathscr{E}_k(\mathbb{F}_q)^{\times \eta_k}.$$

In Algorithm 3 we now generalise Algorithm 2.

Observe that, for each k the computation takes place in different curves (generically at least). Hence every key K_k comes from a different curve. By XOR:ing the d keys K_k we get a single key, obtained from multiple curves.

Algorithm 3: Key generation, multiple curves

```
    Input: Family $\mathscr{E}$ of $d$ elliptic curves; Set of binary numbers $\mathbf{b}$; Set of
           functions $\Phi$; Two ordered multisets of points $\mathsf{P}_{\mathscr{E}}, \mathsf{Q}_{\mathscr{E}} \subset \mathscr{E}(\mathbb{F}_q)$
    Output: Binary numbers $\mathbf{K} := \{\mathsf{K}_k \mid 1 \leq k \leq d\}$
1   for $1 \leq k \leq d$ do
2       Compute the points $\{\mathbf{z}_j^k \mid 1 \leq j \leq \eta_k\} \subset \mathscr{E}_k(\mathbb{F}_q)$ from (4)
3       $\mathsf{K}_k := \mathbf{s}^{\mathrm{L}}((\mathbf{z}_1^k)_x) \bigsqcup \mathbf{s}^{\mathrm{R}}((\mathbf{z}_1^k)_y)$
4       for $2 \leq j \leq \eta_k$ do
5           if $j$ odd then
6               $\mathsf{K}_k := \mathsf{K}_k \bigsqcup \mathbf{s}^{\mathrm{L}}((\mathbf{z}_j^k)_x) \bigsqcup \mathbf{s}^{\mathrm{R}}((\mathbf{z}_j^k)_y)$
7           else
8               $\mathsf{K}_k := \mathsf{K}_k \bigsqcup \mathbf{s}^{\mathrm{L}}((\mathbf{z}_j^k)_y) \bigsqcup \mathbf{s}^{\mathrm{R}}((\mathbf{z}_j^k)_x)$
9   $\mathbf{K} := \{\mathsf{K}_k \mid 1 \leq k \leq d\}$
```

Acknowledgements

Thanks to Kiran Raja for constant pushing and encouragement. He also read a preliminary draft and offered very helpful comments.

References

[1] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *An introduction to mathematical cryptography.* Undergraduate Texts in Mathematics. Springer, New York, second edition, 2014.

[2] Paul Knutson, Kiran Raja, Daniel Larsson, and Raghavendra Ramachandra. Finite field elliptic curve for key generation and biometric template protection. In *2021 IEEE International Workshop on Biometrics and Forensics (IWBF)*, pages 1–6, 2021.

[3] Omar Reyad and Zbigniew Kotulski. Statistical analysis of the chaos-driven elliptic curve pseudo-random number generators. In *Intl. Conf. on Cryptography and Security Systems*, pages 38–48. Springer, 2014.

[4] Joseph H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, 1986.