Conjectured Security of the ANSI-NIST Elliptic Curve RNG

Daniel R. L. Brown[*]

March 24, 2006

Abstract

An elliptic curve random number generator (ECRNG) has been proposed in ANSI and NIST draft standards. This paper proves that, if three conjectures are true, then the ECRNG is secure. The three conjectures are hardness of the elliptic curve decisional Diffie-Hellman problem and the hardness of two newer problems, the x-logarithm problem and the truncated point problem.

Key Words: Random Number Generation, Elliptic Curve Cryptography.

1 Introduction

Certain random number generator (RNG) algorithms, such as the Blum-Micali [BM84] and Kaliski [Kal86] generators, have been proven secure — assuming the conjectured hardness of associated number-theoretic problems. Recently, a new random number generator has been proposed (see [BK05, Joh04]). In this paper, this new generator is called the *Elliptic Curve Random Number Generator (ECRNG)*. Like Kaliski's generator, the ECRNG is based on elliptic curves and is adapted from the Blum-Micali generator. Compared to many other number-theoretic RNGs, the ECRNG is considerably more efficient, because it outputs more bits per number-theoretic operation. This paper proves that ECRNG is secure if the following problems are hard:

- the elliptic curve version of the well known decisional Diffie-Hellman problem (DDHP),
- the x-logarithm problem (XLP), a new problem, which is, given an elliptic curve point, determine whether its discrete logarithm is congruent to the x-coordinate of an elliptic curve point.
- the truncated point problem (TPP), a new problem, which is, given a bit string of a certain length, determine whether it is obtain by truncating the x-coordinate of a random elliptic curve point.

Hardness of the DDHP for certain elliptic curve groups is now widely accepted. The TPP concerns extraction of pseudorandom bits from random elliptic curve points. El Mahassni and Shparlinski [MS02] give some results about extraction of pseudorandom bits from elliptic curve points. Gürel [Gür05] also gives some results, although with fewer bits extracted than in the ECRNG.

The ECRNG is different from the Kaliski RNG in two major respects:

[*]Certicom Research

- The ECRNG uses ordinary elliptic curves, not the supersingular elliptic curves of Kaliski RNG. Supersingular curves can make the state transition function a permutation, which is advantageous for making the Blum-Micali proof work, but is disadvantageous because of the Menezes-Okamoto-Vanstone (MOV) attack, which requires a larger curve to make the proof give a useful assurance[1]. For ordinary curves, the state transition function is many-to-one, often two-to-one. This paper adapts the Blum-Micali proof by introducing the x-logarithm problem to overcome the obstacle introduced by state transition function not being a permutation.

- The ECRNG produces at each state update an output with almost as many bits as in the x-coordinate of an elliptic curve point, whereas the Kaliski ECRNG outputs just a single bit. Therefore the ECRNG is considerably more efficient than the Kaliski RNG if operated over the same elliptic curve. The Kaliski RNG outputs a single bit that is a hardcore predicate for the elliptic curve discrete logarithm problem (ECRNG). The ECRNG output function essentially uses a conjectured hardcore function of the ECDLP. The basis of this conjecture is the elliptic curve DDHP, and the TPP.

2 The Elliptic Curve Random Number Generator

Let $\mathbb{F}_q$ be a finite field with q elements. An elliptic curve E over $\mathbb{F}_q$ is defined by a nonsingular cubic polynomial in two variables x and y with coefficients in $\mathbb{F}_q$. This paper considers only cubics in a specially reduced Weierstrass form

$$E(x,y) = y^2 + cxy - (x^3 + ax^{1+c} + b) = 0 \tag{1}$$

where $c = 1 - q + 2\lfloor q/2 \rfloor$, since these are most often used in cryptography, and particularly in the ECRNG. We define the rational points of the curve to be

$$E(\mathbb{F}_q) = \{(x,y) \in \mathbb{F}_q^2 : E(x,y) = 0\} \cup \{0\}. \tag{2}$$

An addition law is defined on $E(\mathbb{F}_q)$ using the well-known chord-and-tangent law. For example, $(u,v)+(x,y) = (w,z)$ is computed as follows. Form a line through (u,v) and (x,y), which intersects the curve $E(x,y) = 0$ in three points, namely (u,v), (x,y) and some third point $(w,-z)$, which defines the desired sum by negating the ordinate.

In the ECRNG, and in elliptic curve cryptography more generally, one defines some base point P on the curve. One assumes that P has prime order n in the elliptic curve group, so that $nP = 0$. Generally, the number of points in $E(\mathbb{F}_q)$ is hn, where the cofactor h is usually quite small, typically with $h \in \{1,2,4\}$. We say that a point Q is *valid* if it is an additive multiple of P. We will generally only consider valid points in this paper, so when we say a random point, we mean a random valid point.

The ECRNG maintains a state, which is an integer $s_i \in [0, \max(q-1, n-1)]$. The iteration index i increments upon each each output point of the ECRNG. The ECRNG is intended to be initialized by choosing the initial state as s_0 uniformly at random from $\mathbb{Z} \cap [0, n-1]$.

For a point $P = (x,y) \in E(\mathbb{F}_q)$, we write $x(P) = \bar{x}$, where $\bar{x} \in \mathbb{Z}$ is obtained by taking the bit representation of the $x \in \mathbb{F}_q$ and considering this is as the bit representation of an integer. When q is prime, we essentially have $\bar{x} = x$, but when q is not a prime, the value of $\bar{x}$ depends

[1]This is not to say that MOV attack could be applied against the Kaliski RNG for smaller sized curves.

on the representation used for the finite field $\mathbb{F}_q$. (We may arbitrarily define $x(0) = 0$, but we will encounter this case negligibly often in our analysis.) Therefore, to fully specify the ECRNG, one needs to define a field representation, because the function $x(\cdot)$ has an important rôle in the ECRNG, as we see below.

The ECRNG has another initialization parameter, which is a point Q. The point should ideally be chosen at random, preferably verifiably at random, such as by deriving it from the output of secure hash function or block cipher.

When the state is s_i, the (raw) output point is defined as

$$R_i = s_i Q. \tag{3}$$

The actual output of the ECRNG applies further processing to R_i. The final output is $r_i = t(x(R_i))$, where t is a function that truncates certain bits from the bit string representation of an elliptic curve point. The purpose of t is to convert the x-coordinate of a pseudorandom EC point to a pseudorandom bit string. This paper simply conjectures that the function t has the property that for a pseudorandom point R and the output bit string $r = t(x(R))$ is pseudorandom. The main contribution of this paper is to analyze the pseudorandomness of raw output points R_i of the ECRNG.

After generating an output point R_i, the state is updated as

$$s_{i+1} = x(s_i P). \tag{4}$$

It is convenient to adopt the following notation. We define the *prestate* at iteration $i + 1$ as $S_{i+1} = s_i P$. Note that $s_{i+1} = x(S_{i+1})$. We may think of the prestate being updated as

$$S_{i+2} = x(S_{i+1})P. \tag{5}$$

The following notation for the ECRNG will be convenient. Let s_0 be the initial state. Then define functions g_m such that:

$$g_m(Q, s_0) = (R_0, R_1, \ldots, R_m). \tag{6}$$

These functions are both deterministic and efficient. They are related to each other as follows:

$$g_m(Q, s_0) = (s_0 Q, g_{m-1}(Q, x(s_0 P))), \tag{7}$$

where this notation indicates the sequence output by g_{m-1} is prepended by a point to give the sequence output by g_m. That is, this paper uses the convention that a comma indicates concatenation of point sequences.

3 Lemmas on Indistinguishability

Random variables X and Y are *computationally indistinguishable* if an adversary that is given a sample value u that has probability $\frac{1}{2}$ of coming from X and $\frac{1}{2}$ from Y, the adversary cannot distinguish[2], with a feasible cost of computation and reasonable success rate, whether u comes from X or from Y. Pseudorandomness is indistinguishability from a uniform (equiprobable) distribution. Indistinguishability is a well known notion in cryptology, but for completeness, this section

[2]One might, like the author did, first think that distinguishing between X and Y is the problem of being given two values, one from X and one from Y and determining which is which. This kind of indistinguishability (left-or-right) may be written as $(X, Y) \sim (Y, X)$, and can easily be shown to be equivalent to $X \sim Y$ (real-or-fake).

introduces some general notation and lemmas on indistinguishability that are convenient for the proofs of the main theorems.

We write $X \sim Y$ to indicate that random variables X and Y are indistinguishable. Where needed, we write $X \overset{\sigma}{\sim} Y$ to quantify the indistinguishability by some parameters σ, such as success rate or computational cost. We write $X \cong Y$ when random variables X and Y are identically distributed. Obviously, $X \cong Y$ implies $X \sim Y$.

Intuitively, one expects indistinguishability ($\sim$) to be an equivalence relation. Certainly, $\sim$ is reflexive and symmetric, and more interestingly, it is transitive ([Lub96, Ex. 27], for example). This is such a fundamental point it is worth repeating here.

Lemma 1. *If $X \sim Y$ and $Y \sim Z$, then $X \sim Z$.*

Proof. Suppose that A is an algorithm that attempts to distinguish X and Z. That is, suppose that A always outputs "X" or "Z". Let x be the probability A outputs "X" when given input from X, let z be the probability that A outputs "Z" when given input Z. Distinguisher A is *successful* if $x + z > 1$.

Let x' and z' be the probabilities that A outputs "X" and "Z" respectively, when given input from random variable Y. Note that $x' + z' = 1$. Let A_x be modification of A where output "Z" is changed to "Y". Because $X \sim Y$, by definition it is true that A_x is not a successful distinguisher between X and Y. Therefore $x + z' \leqslant 1$. Similarly, let A_z be defined as A with output "X" converted to "Y". Because $Y \sim Z$, we have $x' + z \leqslant 1$. Adding these two inequalities we get $x + z' + x' + z \leqslant 2$. Subtracting $z' + x' = 1$ from this gives $x + z \leqslant 1$, which contradicts our original assumption that A distinguishes between X and Z.

More generally, if we only know that $x + z' \leqslant 1 + \alpha_{xy}$ and $x' + z \leqslant 1 + \alpha_{yz}$, where α_{xy} and α_{yz} are the maximum possible *advantage* of distinguishing algorithms between X and Y, and between Y and Z, respectively, then we get $x + z \leqslant 1 + \alpha_{xy} + \alpha_{yz}$, so that $\alpha_{xz} \leqslant \alpha_{xy} + \alpha_{yz}$. In other words, if $X \overset{\alpha_1}{\sim} Y$ and $Y \overset{\alpha_2}{\sim} Z$, then $X \overset{\alpha_1+\alpha_2}{\sim} Z$.

When time, or over all computational cost of the distinguishers is also considered, then note that A, A_x and A_z all have essentially the same cost. Therefore the cost of distinguishing between X and Z is at least the minimum of the costs of distinguishing between X and Y or between Y and Z. □

A second lemma, which one also intuitively expects, makes proofs cleaner through separating complicated constructions from indistinguishability.

Lemma 2. *If f is efficiently computable functions, and $X \sim Y$, then $f(X) \sim f(Y)$.*

Proof. Suppose that $f(X)$ and $f(Y)$ were distinguished by algorithm A. Then X and Y can be distinguished by applying f to a challenge value u, then applying A to $f(u)$. The advantage of distinguishing is the same, but the computational cost is that of A plus that of f. □

It is worth noting that the converse to this lemma does not necessarily hold: generally, $f(X) \sim f(Y)$ may not imply $X \sim Y$. Trivial counterexample include f a constant function. Nontrivial counterexamples exist too, such as f being a bijection whose inverse is not efficiently computable.

A third lemma, which one again intuitively expects, allows one to build indistinguishable distributions with more variability and to analyze distributions into independent components.

Lemma 3. *If $X \sim Y$ and $W \sim Z$, and X and W are independent variables, as are Y and Z, and that X and Z can be efficiently sampled, then $(X, W) \sim (Y, Z)$.*

Proof. We claim that if $X \sim Y$, then $(X, Z) \sim (Y, Z)$. To see this, suppose that algorithm A distinguishes (X, Z) from (Y, Z). Suppose that u is a sample of X or Y, and we wish to distinguish which is its source, X or Y. Take a random sample z from Z and apply A to (u, z). If A says (u, z) is from (X, Z), then we say that u is from X, and vice versa for Y. Similarly, we can show that if $W \sim Z$, then $(X, W) \sim (X, Z)$ by using a sample from x. Therefore $(X, W) \sim (X, Z) \sim (Y, Z)$, so transitivity gives $(X, W) \sim (Y, Z)$. The advantages add, and the costs only change by the cost sampling X and Z. $\square$

This lemma also applies under our notational convention that if X and Y are sequences, then (X, Y) is their concatenation.

4 The Decisional Diffie-Hellman Problem

The *decisional Diffie-Hellman problem (DDHP)* for a given elliptic curve E and a base point P is to distinguish between a triple $(Q, R, S) = (qP, rP, qrP)$ and a triple $(Q, R, Z) = (qP, rP, zP)$ where q, r, z are integer random variables uniformly distributed in the interval $[0, n-1]$. (Note this q is not to be confused with the field order.) The triple (Q, R, S) is often called a Diffie-Hellman triple. For certain elliptic curves, it is conjectured that DDHP is a hard problem.

Conjecture 1. *If q, r and z are independent random integers uniformly distributed in $[0, n-1]$, then $(qP, rP, qrP) \sim (qP, rP, zP)$.*

This provides a nontrivial counterexample to the converse to Lemma 2 because random variables $X = (q, r, qr)$ and $Y = (q, r, z)$ are distinguishable, but if one applies the function f defined by $f(x, y, z) = (xP, yP, zP)$, then the conjecture says $f(X) \sim f(Y)$.

The conjectured hardness of the DDHP, for certain groups, is widely believed among cryptologists. One should be aware that for certain elliptic curves, however, there exists a function, called a pairing that potentially distinguishes Diffie-Hellman triples. Pairings exist for all elliptic curves, but only for a very few are they known to be efficient. For most elliptic curves, one can verify that the known pairings are extremely inefficient and infeasible to use in practice. This has been confirmed for most of the NIST recommended elliptic curves.

5 The x-Logarithm Problem

The *x-Logarithm Problem (XLP)* for elliptic curve $E(\mathbb{F}_q)$ and base point P is to distinguish between dP and xP where: d is an integer chosen uniformly at random in $[0, n-1]$; and $x = x(Z)$ for a point Z chosen uniformly at random in $E(\mathbb{F}_q)$. We conjecture that the x-logarithm problem is hard for most elliptic curves:

Conjecture 2. *If d and z are random integer uniformly distributed in the interval $[0, n-1]$, then $dP \sim x(zP)P$.*

Now d and $x = x(zP)$ are generally distinguishable. Firstly, known tests on x quickly determine whether there exists a $y \in \mathbb{F}_q$ such that $(x, y) \in E(\mathbb{F}_q)$. Secondly, when the cofactor $h > 1$, we have $x > n$ for at least about half of the x-coordinates x of random points, which is not true for d. Therefore, this conjecture gives another counterexample to the converse of Lemma 2.

An intuitive reason for the plausibility of the XLP conjecture is that given public key dP, one expects that nothing substantial is leaked about the private key d. This intuition derives from the conjectured hardness of the elliptic curve discrete logarithm problem (ECDLP). However, a formal argument that the ability to determine whether $dP = x(zP)P$ for some z, implies an ability to find d is not known to the author. In fact, conceivably, the ECDLP could be hard, even though certain information about the discrete logarithm, such as whether it is congruent to an x-coordinate, is easily discernible.

Certain bits in the binary representation of the d have been shown by Kaliski [Kal86] to be as hard to find as the whole of d. A bit of information with such a property are known are *hardcore predicate* for the ECDLP. Conjecture 2 is therefore that the bit of information whether or not the discrete logarithm is congruent to an x-coordinate is a *hardcore predicate* for the ECDLP. Kaliski's proof that certain bits of the binary representation of the discrete logarithm are hardcore predicate works with a reduction, as follows. Given $Q = dP$, determine from Q a bit of information about d. Then transform Q to some $Q' = d'P$ in such a way that there is an known relation between d and d', and d' has one less bit of freedom than d. Next determine a bit of information about d', then d'' and so on. The transformation is such that all bits of information learnt are independent and can be easily be reconstituted to learn d in its entirety. What would be ideal to make Conjecture 2 into a theorem, would be another transformation with comparable properties to Kaliski's for determining the discrete logarithm d using an oracle for solving the XLP.

6 Indistinguishability of the ECRNG Outputs Points

The traditional notion of security for an RNG is that its output is indistinguishable from random. We prove below in Theorem 4 that the raw output points of the ECRNG are indistinguishable from random points, that they are pseudorandom as points.

The following proof is not substantially different than the proof for the Blum-Micali generator [BM84] or the Kaliski generator [Kal86]. However, unlike these generators, which used a hardcore bit of the discrete logarithm, the ECRNG uses a hardcore function — as suggested, for example, in [Gol01] — which yields greater efficiency provide that one accepts hardness of the corresponding problem, the DDHP to ensure the function is hardcore. A second reason for providing the proof anew here is that the state update transition function is not a permutation. This issue is addressed via recourse to the hardness of the x-logarithm problem. Roughly speaking, hardness of the XLP ensures that the state transition function is indistinguishable from a permutation.

A stronger security notion, *forward secrecy*, is now required in the ANSI and NIST draft RNG standards. The forward secrecy of the ECRNG is proved in Theorem 5 in §7, and implies Theorem 4 below. In fact, the presentation of the proof Theorem 5 happens to be simpler than the proof below of Theorem 4 for three reasons: the proof below is more explicit in notation and use of the previous lemmas, the proof below works from left-to-right rather than from right-to-left, and the proof below works directly on the stated claim rather than a stronger claim that makes the proof simpler but is otherwise unmotivated. Therefore, the reader may wish to skip to the proof Theorem 5 in §7, for a simpler proof presented a higher level with less explicit details.

Theorem 4. *If the DDHP and XLP are hard, and $Q, Z_0, \ldots, Z_m$ are independent and uniformly distributed random points, and s_0 is a random integer uniformly distributed in $[0, n-1]$, and $g_m(Q, s_0) = (R_0, \ldots, R_m)$, then*

$$(Q, R_0, \ldots, R_m) \sim (Q, Z_0, \ldots, Z_m). \tag{8}$$

Proof. The basic idea is to first replace R_0 by Z_0, using assumed hardness of the DDHP, and then proceed by induction on m for the remaining the points. The proof uses the lemmas about indistinguishability previously established. Again, the assumed hardness of the DDHP is invoked to show that each output points is indistinguishable from a random points if the prestate points are random points. The assumed hardness of the XLP is invoked to ensure the that when the state is updated, the fact that it is now restricted to an x-coordinate is not of consequence.

Let q, s, z be independent random integers uniformly distributed in $[0, n-1]$. Let $(Q, R, S) = (qP, qsP, sP)$ and let $(Q, R, Z) = (qP, qsP, zP)$. Writing $r = qs$, and $s = q/r$, working modulo n, we clearly have $(Q, R, S) \sim (Q, R, Z)$ by hardness of the DDHP. Define a function

$$f_m(A, B, C) = (A, B, g_{m-1}(A, x(C))). \tag{9}$$

Because $s \cong s_0$, it follows that $(Q, R_0, R_1, \ldots, R_m) = (Q, g_m(Q, s_0)) \cong (Q, g_m(Q, s))$. By definition, we have $(Q, g_m(Q, s)) = (Q, sQ, g_{m-1}(Q, x(sP))) = f_m(Q, sQ, sP) = f_m(Q, R, S) \sim f_m(Q, R, Z)$, with the last step done by applying Lemma 2 to $(Q, R, S) \sim (Q, R, Z)$. Therefore, by transitivity

$$(Q, R_0, \ldots, R_m) \sim f_m(Q, R, Z). \tag{10}$$

Continuing, $f_m(Q, R, Z) = (Q, R, g_{m-1}(Q, x(Z))) \cong (Q, Z_0, g_{m-1}(Q, x(Z)))$, because $R \cong Z_0$ and R and Z_0 are independent from Q and Z.

Having used the DDHP, we will next use the XLP to get an inductive step. By induction we have, $(Q, g_{m-1}(Q, s_0)) = (Q, R_0, \ldots, R_{m-1}) \sim (Q, Z_0, \ldots, Z_{m-1})$, and therefore $g_{m-1}(Q, s_0) \sim (Z_0, \ldots, Z_{m-1})$. Clearly $(Z_0, \ldots, Z_{m-1}) \cong (Z_1, \ldots, Z_m)$, so for any $u \cong s_0$, we have $g_{m-1}(Q, u) \sim (Z_1, \ldots, Z_m)$. We claim that

$$g_{m-1}(Q, u) \sim g_{m-1}(Q, x(Z)). \tag{11}$$

Now $(Q, g_{m-1}(Q, u)) = f_{m-1}(Q, uQ, uP)$ and $(Q, g_{m-1}(Q, x(Z))) = f_{m-1}(Q, x(Z)Q, x(Z)P)$, so it suffices to prove that $(Q, uQ, uP) \sim (Q, x(Z)Q, x(Z)P)$. Define a function $h(a, B) = (aP, aB, B)$. Then $(Q, uQ, uP) = h(q, uP)$ and $(Q, x(Z)Q, x(Z)P) = h(q, x(Z)P)$. Now $uP \sim x(Z)P$ by the hardness of the XLP, so $(q, uP) \sim (q, x(Z)P)$, and thus $h(q, uP) \sim h(q, x(Z)P)$, which proves (11).

Therefore

$$\begin{aligned}(Q, R_0, \ldots, R_m) &\sim f_m(Q, R, Z) \\ &\sim (Q, Z_0, g_{m-1}(Q, x(Z))) \\ &\sim (Q, Z_0, g_{m-1}(Q, u)) \\ &\sim (Q, Z_0, Z_1, \ldots, Z_m),\end{aligned} \tag{12}$$

so transitivity gives the final result. □

This proof makes essential use of Q being random. The reason for this is more than just to make the proof work. If Q is not random, then it may be the case the adversary knows a d such that $dQ = P$. Then $dR_i = dS_{i+1}$, so that such a distinguisher could immediately recover the secret prestates from the output. Once the distinguisher gets the prestates, it can easily distinguish the output from random. Therefore, it is generally preferable for Q to be chosen randomly, relative to P.

Although Theorem 4 says that hardness of the DDHP is one of the *sufficient* conditions for indistinguishability of the ECRNG output points, it is not at all clear whether or not hardness of the DDHP is a *necessary* condition. It is clear that hardness of the *computational* Diffie-Hellman problem (CDHP) is a necessary condition in that S_{i+1} is the Diffie-Hellman product of P and R_i to the base Q. Therefore, one can reasonably hope to significantly strengthen Theorem 4 by proving that hardness of the CDHP is one of a set of sufficient conditions.

In contrast, hardness of the XLP is necessary for indistinguishability of the raw output points. Output $R_1 = s_1P = x(S_1)Q$, so distinguishing it from random Z_1 is essentially the XLP. Distinguishing output $R_j = x(S_{j-1})P$ from random is almost the XLP except that S_{j-1} is not necessarily a random point. However, if distinguishing algorithm A is an XLP solver, then one heuristically expects that A could distinguish R_j from a random point. Algorithm A would only fail if the S_{j-1} were distributed with a bias such that A reports that $x(S_{j-1})P$ was not of the form $x(Z)P$ for some valid point Z. Therefore one cannot hope to strengthen Theorem 4 by replacing the hardness of the XLP with a weaker yet still natural assumption. One could improve the result, however, by proving the the XLP is as hard as some other problems, such as the DDHP or ECDLP.

Although hardness of the XLP is necessary for the raw output points to be pseudorandom, it does not seem necessary for the full ECRNG output bit strings to be pseudorandom. Likewise, hardness of the CDHP may not be necessary for security of the full ECRNG, even it is necessary of the indistinguishability of the raw output points. Truncation of the raw output points may yield bit strings are that unusable even by an XLP distinguisher or a CDHP solver to distinguish the ECRNG outputs.

7 Forward Secrecy (Backtracking Resistance)

In cryptology, *forward secrecy* refers to the following property: present secrets remain secret into the future, even from an adversary who acquires all future secrets. So, in forward secrecy, the secrecy of present secrets extends forward into the future indefinitely and without depending on protection of some future secrets. Incidentally, in ANSI X9.82 and NIST SP 800-90, forward secrecy has been retermed[3] *backtracking resistance* to convey the notion that an adversary cannot use future secret to backtrack to present secrets.

Many key agreement schemes, and even some digital signature schemes, claim forward secrecy. When implementing these schemes, however, one may need to ensure that the RNG used also has forward secrecy, otherwise the forward secrecy of the key agreement scheme may be undermined by the RNG.

Theorem 4 is now strengthened to provide forward secrecy. We let adversary see the latest prestate, but still it cannot distinguish previous output points from random points.

Theorem 5. *If the DDHP and XLP are hard, and $Q, Z_0, \dots, Z_m, Z_{m+1}$ are independent and uniformly distributed random points, and s_0 is a random integer uniformly distributed in $[0, n-1]$, and $g_m(Q, s_0) = (R_0, \dots, R_m)$, with the next prestate of the ECRNG being S_{m+1}, then*

$$(Q, R_0, \dots, R_m, S_{m+1}) \sim (Q, Z_0, \dots, Z_m, Z_{m+1}). \tag{13}$$

Proof. The case of $m = 0$ is to show $(Q, R_0, S_1) \sim (Q, Z_0, Z_1)$. This follows directly from hardness of the DDHP. Assume by induction that

$$(Q, R_0, \dots, R_{m-1}, S_m) \sim (Q, Z_0, \dots, Z_{m-1}, Z_m). \tag{14}$$

The current outputs and prestate are given by

$$(Q, R_0, \dots, R_{m-1}, R_m, S_{m+1}) = (Q, R_0, \dots, R_{m-1}, x(S_m)Q, x(S_m)P) \tag{15}$$

[3]Breaking precedent not only with wider usage in cryptology but also with other ANSI standards such as X9.42 and X9.62, which use forward secrecy.

Combining (14) and (15), we get

$$(Q, R_0, \ldots, R_{m-1}, R_m, S_{m+1}) \sim (Q, Z_0, \ldots, Z_{m-1}, x(Z_m)Q, x(Z_m)P). \tag{16}$$

Hardness of the XLP gives $x(Z_m)P \sim Z_{m+1}$. Writing $Q = qP$, Lemma 2 gives

$$(Q, Z_0, \ldots, Z_{m-1}, x(Z_m)Q, x(Z_m)P) \sim (qP, Z_0, \ldots, Z_{m-1}, qZ_{m+1}, Z_{m+1}). \tag{17}$$

Hardness of the DDHP gives $(qP, qZ_{m+1}, Z_{m+1}) \sim (Q, Z_m, Z_{m+1})$ where $Q = qP$, so Lemma 3 gives

$$(qP, Z_0, \ldots, Z_{m-1}, qZ_{m+1}, Z_{m+1}) \sim (Q, Z_0, \ldots, Z_{m-1}, Z_m, Z_{m+1}). \tag{18}$$

Lemma 1 on transitivity connects (16) to (17) to (18) to complete the inductive step, getting us our desired result. □

Recapping, an adversary who captures the current state cannot use it to distinguish previous outputs from random, and consequently cannot determined any previous outputs. Conversely, the an adversary cannot use the previous outputs to learn anything about the current state.

Theorem 5 immediately implies Theorem 4 just by dropping the last point. Therefore, Theorem 4 has been proven twice. (The proofs are not quite equivalent, since the first proof does induction by growing points at the beginning of the sequence, while the second does so by growing points at the end.)

8 Truncated Point Problem and Security of the Full ECRNG

For appropriate choice of truncation function $t(\cdot)$, we conjecture the following.

Conjecture 3. *Let R be a random point and b a random bit string of length matching the output length of $t(\cdot)$. Then $t(x(R)) \sim b$.*

We call the problem of distinguishing between $t(x(R))$ and b, the *Truncated Point Problem (TTP)*. This paper does not substantially address this conjecture, but rather uses to prove something about the final output ECRNG, rather than just its raw output points.

Theorem 6. *If DDHP, XLP and TTP are hard, then the ECRNG has forward secrecy.*

Proof. Apply Theorem 5 to get that the raw outputs are indistinguishable. By the assumed hardness of the TTP, each truncated point is indistinguishable from random bit strings. Apply the lemmas as necessary and get that the ECRNG output bit strings are indistinguishable from random bit strings, even from an adversary that gets to see the latest prestate. □

The proposed truncation function drops some number of the leftmost bits of the bit representation of the x-coordinate. The number of bits dropped is at least $13 + \log_2(h)$, where h is the cofactor. The number of bits dropped must also be such that resulting length is a multiple of eight. Current draft standards currently allows any number of bits to be dropped that meets these conditions.

It should be noted that for the NIST recommended curves defined over the binary field $\mathbb{F}_{2^{409}}$, valid elliptic curve points have a fixed rightmost bit in their canonical representation. Therefore, for the curves B-409 and K-409, the truncation function should also drop the rightmost bit. The explanation for this phenomenon is that one of the conditions for a point to have the correct order can be characterized by the trace of the x-coordinate have a fixed value. The trace depends on the field representation. For trinomial and pentanomial field representations, the trace simplifies to a sum of just a few of the bits, the *trace bits*, in the representation. In all fields, the rightmost bit is a trace bit. For the 409-bit field, the rightmost bit is the only trace bit. For the other four NIST recommended binary fields, there is at least one trace bit among the leftmost truncated bits. Consequently, the constant trace condition does not leak any information after truncation in these cases.

9 Initialization, Reseeding and Prediction Resistance

The ANSI and NIST draft standards specify further details of the ECRNG that have been omitted from in this paper. These further details generally involve ensuring that the state value s_i is initialized or refreshed with an adequate quantity of entropy. This paper has assumed, for simplicity, that the initial state s_0 is arranged to have a uniform distribution, which is secret to all adversaries.

This paper has not attempted to analyze the various issues surrounding entropy of the secret state. Prediction resistance is the ability of RNG to add additional entropy into the secret state to recover completely from a circumstance where an adversary knows information about the previous state. Initialization and prediction resistance are general issues to any RNG, and indeed the draft standards specifying the ECRNG do not treat the ECRNG especially different from other RNGs with respect initialization and prediction resistance. This paper deliberately restricts itself to ECRNG specific issues.

10 Conclusion

The ECRNG has both proven number-theoretic-based security and higher efficiency than many other RNGs with similar security properties. Random numbers play such an essential in cryptography, that implementers should always choose them to be as secure as possible. Therefore, the ECRNG should be a serious consideration, and its high efficiency makes it suitable even for constrained environments.

Acknowledgments

I thank the ANSI X9F1 working group for introducing me to the ECRNG, and Certicom colleagues for valuable discussions.

References

[BK05] Elaine Barker and John Kelsey, *Recommendation for random number generation using deterministic random bit generators*, Special Publication 800-90, National Institute of Standards and Technology, December 2005, Draft `http://csrc.nist.gov/publications/drafts/sp800-90_draft_dec2005.pdf`.

[BM84] Manuel Blum and Silvio Micali, *How to generate cryptographically strong sequences of pseudo-random bits*, SIAM Journal on Computing **13** (1984), 850–864.

[Gol01] Oded Goldreich, *Foundations of cryptography*, Cambridge University Press, 2001.

[Gür05] Nicolas Gürel, *Extracting bits from coordinates of a point of an elliptic curve*, Cryptology ePrint Archive, Report 2005/324, 2005, `http://eprint.iacr.org/`.

[Joh04] Don B. Johnson, *X9.82 part 3 number theoretic DRBGs*, Presentation at NIST RNG Workshop, July 2004, `http://csrc.nist.gov/CryptoToolkit/RNG/Workshop/NumberTheoreticDRBG.pdf`.

[Kal86] Burton S. Kaliski, *A pseudo-random bit generator based on elliptic logarithms*, Advances in Cryptology — CRYPTO '86 (A. M. Odlyzko, ed.), LNCS, vol. 263, 1986, pp. 84–103.

[Lub96] Michael Luby, *Pseudorandomness and cryptographic applications*, Princeton University Press, 1996.

[MS02] Edwin El Mahassni and Igor Shparlinksi, *On the uniformity of distribution of congruential generators over elliptic curves*, International Conference on Sequences and Their Applications, SETA '01, 2002, pp. 257–264.

A Unpredictability of the Next State from the Current Output

Unpredictability is a much weaker property than indistinguishability, but is also much more important. If the ECRNG outputs are used as cryptographic keys, very little harm may come from them being distinguishable. If they are predictable, however, then all may be lost. Indistinguishability implies unpredictably, so in fact, we have already proven unpredictability.

The theorem below, however, proves a little bit of unpredictability under weaker, arguably more accepted, conjectures, such as hardness of the CDHP instead of the hardness of the DDHP.

Theorem 7. *If the CDHP and XLP are hard, and q and s_0 are independent random integers uniformly distributed in $[0, n-1]$, and $g_m(qP, s_0) = (R_0, \ldots, R_m)$ and $Q = qP$, then an adversary who gets to see only Q and R_m cannot compute the next prestate S_{m+1}.*

Proof. Clearly $S_1 \sim Z$ where $Z = zP$ and z is a random integer uniformly distributed in $[0, n-1]$. Indeed, $s_0 \cong z$, so $S_1 = s_0P \cong zP = Z$. Assume by induction that $S_{j-1} \sim Z$. Now $S_j = x(S_{j-1})P \sim x(Z)P \sim Z$, with the second indistinguishability flowing from the hardness of the XLP. Therefore $S_{m+1} \sim Z$. Since q is independent of z, we have $(Q, S_{m+1}) \sim (Q, Z)$. Now $(Q, R_m) = (Q, qS_{m+1}) \sim (Q, qZ) \sim (Q, Z)$, with the second indistinguishability flowing from Z being able to absorb q by independence.

Suppose adversary A takes (Q, R_m) and outputs $S_{m+1} = q^{-1}R_m$. Then adversary can also take (Q, Z) and output $U = q^{-1}Z$, because otherwise A could distinguish (Q, R_m) from (Q, Z). Let $(X, Y) = (xP, yP)$ with x, y independent random integers uniformly distributed in $[0, n-1]$. We will use A to compute xyP. Pick a random integer u with the same distribution. Let $U = uP$. Apply A to (X, U) to get $V = x^{-1}U = x^{-1}uP$. Let $W = u^{-1}V = x^{-1}P = wP$. Apply A to (W, Y) to get $w^{-1}Y = (x^{-1})^{-1}Y = xY = xyP$, as desired. Because we assumed that CDHP is hard, adversary A cannot find xyP, so we get a contradiction. □

The simple proof techniques above do not seem to rule out an adversary who could use two output points to find the next state, or one output point to find the next output point. The obstacle in the first case seems to be that output points obey a relationship that needs to be simulated if we wish to solve the CDHP. The obstacle in the second case is that the next output can be thought of as a one-way function of the Diffie-Hellman product of public values, and we seem to need to invert it to solve the CDHP.