DEFCON 33 - 0x3F "? Cube" - Modern Cryptography Track Writeup

1nfocalypse

August 13, 2025

"Introduction to that thing, but only for people who already know it."

Introduction

The Cube, or "? Cube", is a multifaceted team-based CTF offered at DEFCON. It has been a black badge challenge for two consecutive years, featuring lockpicking, exploitation, analysis, cryptography, and more. This year's offering contained 5 challenges in Modern Cryptography. This document discusses the motivation behind each, the mathematics behind each system, and the intended solution path from the designer's point of view. It also lists references for further information on each attack or for the creation of the challenge specifically. If you are reading this after competing, I hope that you found the challenges interesting! While nobody was able to solve all of them, I hope that this document will be educational for those curious. Thank you for playing, and good luck next year!

Challenge 1: Taking Candy From the Crib

This challenge is an exercise in breaking a weak stream cipher. The player is offered ciphertext, some partial information about the cleartext, and a hint as to what form the keystream generator takes. With this, they are tasked to cribdrag the information they know about the cleartext out of the ciphertext, and use the keystream to invert the generator, in turn enabling them to fully decrypt the ciphertext, revealing the flag.

The keystream generator used is a Linear Congruential Generator, more specifically the $ranqd1$ procedure from Numerical Recipes. This is a LCG of the form:

$$o_{n+1} = (a * o_n + c) \bmod m$$

Furthermore, it has additional structure in that $m \in \{2^n | n \in \mathbb{N}\}$ and satisfies the Hull-Dobel Theorem:

- GCD$(m, c) = 1$
- $a - 1$ is divisible by the prime factors of m
- $a - 1$ is divisible by 4 if m is divisible by 4

This, in turn, means that it is a maximal-period LCG, and highly performant relative to the other maximal period variant (m is prime).

It is not lagged in any way, meaning that 5 consecutive outputs is enough to begin inverting the generator. This comes from a linear structure within the outputs, where they will lie on (hyper)planes when plotted in higher dimensions. A genericized walkthrough of this challenge is as follows.

Given a ciphertext c constructed by the relation $p \oplus s$, where p is the plaintext and s is the keystream, the player must first treat what they know about the plaintext (in this case, it is a pastebin link, so of the form https://pastebin.com/) as a bitstring. Given that the LCG generates 32 bit outputs, we can convert the partially-known plaintext to a binary representation and perform a XOR operation to obtain the keystream. Specifically, we obtain http s:// past ebin .com, which enables us to access five outputs of the generator, $O = \{o_1, o_2, ..., o_5\}$. Given the players do not know that it is ranqd1 generating the stream, to invert, they must first deduce the modulus.

To deduce the modulus of a LCG, one needs at least five consecutive outputs of the LCG. They must then create two pseudo-Hankel matrices of the form:

$$\begin{bmatrix} o_n & o_{n+1} & 1 \\ o_{n+1} & o_{n+2} & 1 \\ o_{n+2} & o_{n+3} & 1 \end{bmatrix}$$

With o_n being o_1 and o_2, respectively. Once these matrices are constructed, one solves for the determinants D_1, D_2, and finds the GCD of them as $m = GCD(D_1, D_2)$. Because of the linearity in outputs, this result is a multiple of the modulus, or the modulus itself. In the case of ranqd1, it is a multiple, and given some prior knowledge or ability to research about inverting LCGs, it is quite obvious that the modulus is 2^{32}. As such, we have enumerated:

$$o_{n+1} = (a * o_n + c) \bmod 2^{32}$$

From here, we now must find a, c. To find a, we first must consider if $GCD(m, o_2 - o_1) = 1$. If this is true (as it indeed is with this challenge), we have a solution for a as $(o_3 - o_2)(o_2 - o_1)^{-1}$. If it is not true, it implies that the generator was selected with poor parameters, however, solving for a requires solving a Linear Diophantine Equation. For those curious about this case, a thorough treatment of it can be found in my paper "Cracking Chaos". We will now proceed to determining the additive value c. Set up the equation:

$$(o_1 * a + c) \equiv o_2 \bmod 2^{32}$$

Then, simply solve for c. This yields the complete form of the generator, in turn enabling for additional outputs, allowing for full decryption of the flag.

Challenge 2: Friendly Primes

This is a very simple RSA challenge meant to be a quick pick-me-up after Challenge 1. However, it is a real vulnerability that has been encountered in the past, notably with FujiFilm printers and other embedded/constrained systems. It is an exercise in identifying and breaking moduli generated from poorly selected prime numbers. In our case, we establish a probable prime p with Miller-Rabin, and increment by two until we find another nearby prime q. Players are presented an RSA-4096 public key generated from p, q, and tasked with performing a hostile factorization to decrypt a message containing the flag. While at first quite intimidating, the solution can be found very quickly by attempting Fermat Factorization. This works in the case that $p \approx q$. The general concept of the algorithm is to take the square root of N and explore the immediate area for numbers $p, q : p * q = N$. While not useful for factorizing a properly generated RSA modulus, $p \not\approx q$, it is incredibly fast in this case. A description of the algorithm is as follows:

Algorithm 1 Fermat Factorization

```
1: procedure FermatFactor($N, t$)
2:     tries := t
3:     a := isqrt(n)
4:     c := 0
5:     while not isSquare($a^2 - N$) do
6:         a := a + 1
7:         c := c + 1
8:         if c > tries then
9:             return False
10:        end if
11:    end while
12:    bsq := $a^2$ - $N$
13:    b := isqrt(bsq)
14:    p := a + b
15:    q := a - b
16:    return p, q
17: end procedure
```

Challenge 3: Hasty Handshakes

Another RSA challenge, this one is also not meant to be particularly difficult. It utilizes an attack called Håstad's Broadcast Attack, which is a simple case of Coppersmith's Attack. While not difficult, this challenge is meant to frustrate those unfamiliar with RSA cryptanalysis before they are able to jump into the (significantly) harder elliptic curve challenges later on. The challenge is presented as three (different) RSA-2048 public keys, all with a low encryption exponent ($e = 3$). They are accompanied by three ciphertexts, one from each public key. To make this a bit more difficult, it is not directly stated to the players that they encrypt the same plaintext. Given how prolific this attack is, it is presupposed that it should be recognized from this setup. A description of Håstad's attack is below.

Suppose an unknown, but identical message M is encrypted with numerous RSA public keys, all using the same (small) encryption exponent e. We further specify that each public key has a different modulus N, such that the resulting ciphertexts are not the same, and furthermore that for each modulus N_i, $GCD(N_i, N_j) = 1$ with the corresponding moduli $N_{j \neq i}$. Given three such ciphertexts $C_{1,2,3}$, we turn to the Chinese Remainder Theorem (CRT). The CRT states that, given coprime numbers $n_1, ..., n_k : n_i > 1$ (our moduli) and the product $N = \prod n_i$, along with integers $a_1, ..., a_k$ (our ciphertexts) such that $\forall a_i, 0 \leq a_i < n_i$, there is a unique integer $x \in \frac{\mathbb{Z}}{N\mathbb{Z}}$ such that the remainder of Euclidean division of x by n_i is a_i. More concisely, we have:

$$\frac{\mathbb{Z}}{N\mathbb{Z}} \cong \frac{\mathbb{Z}}{n_1\mathbb{Z}} \times ... \times \frac{\mathbb{Z}}{n_k\mathbb{Z}}$$

Given this, we deduce we are able to compute a number $C \equiv C_i \bmod N_i$. Thus, we find $C \equiv M^3 \bmod N_1 * N_2 * N_3$. Since $M < N_i, \forall N_i$, we find $M^3 < N_1 N_2 N_3$, in turn allowing for computation of the cube root of C, giving us M, which is the flag.

Challenge 4: Cracking Crypto Like the NSA

This is the first elliptic curve challenge of the track, and is very implementation-heavy. However, it is also quite direct. Players are tasked with reconstructing DUAL_EC_DRBG, a backdoored CSPRNG developed by the NSA as a part of their Bullrun program. To make things even more ironic, they are tasked to do so using DJB's curve25519. This was done not only because it's funny, but because it showcases that even with a secure and very well-known curve, this backdoor is still completely functional. Once this is done, they must utilize the backdoor (with the secret either offered as a solution to another challenge or directly given) to deduce the key used to encrypt the flag with AES-256-GCM. They are given the ciphertext, the nonce, and the implementation data for Curve25519 + the DUAL_EC_DRBG variant used to generate the numbers. Since there really isn't a trick to this, we will take this section to review elliptic curve cryptography and the generator itself.

An elliptic curve is a curve $\mathcal{E}$ of the form $y^2 = x^3 + ax + b$, with the additional condition that $4a^3 + 27b^2 \neq 0$. This form is known as a Weierstrass form, and is very typical to see in literature (despite other forms discussed later being far more prolific in application). For cryptographic purposes, we typically define these curves over finite fields, that is, modulo some prime, represented as $\mathcal{E}(\mathbb{F}_q)$. We additionally differentiate points on the curve as a full group structure - that is, under a special definition of addition, we hold the following axioms:

- Identity: $\exists i : \forall a \in G, a + i = a$
- Closure: $\forall a, b \in G, a + b \in G$
- Associativity: $\forall a, b, c \in G, (a + b) + c = a + (b + c)$
- Inverse: $\forall a \in G, \exists a^{-1} \in G : a + a^{-1} = i$

In the context of elliptic curves, we have the identity as $\mathcal{O}$, or the point at infinity. We additionally have a fifth axiom:

- Commutativity: $\forall a, b \in G, a + b = b + a$

Making this a further refinement called an Abelian group (which is what allows things like Elliptic Curve Diffie-Hellman, or ECDH, to work). We can, more specifically, define three further operations as follows:

- Point Addition: $P + Q$
- Point Doubling: $P + P$
- Scalar Multiplication/Exponentiation: $kP : \underbrace{P + P + \cdots + P}_{k \text{ times}}, k \in (0, ord(P))$

The latter is typically the most widely used in Elliptic Curve Cryptography to form the basis of the Elliptic Curve Discrete Logarithm Problem, which states: Given $P, Q \in \mathcal{E}(\mathbb{F}_q), Q : Q = kP$ for an unknown scalar k, determine k. This is presumed hard in a classical context. We will now move to the case of Montgomery Curves, which is the class of curve that curve25519 is. We will also informally discuss why curve25519 is safer as a result of being a Montgomery curve. Montgomery curves enable a specific form of x-coordinate only arithmetic via the Montgomery Ladder - these operations take place in the projective plane, i.e. utilization of (X, Y, Z) coordinates. This, in turn, allows for a constant-time, constant-op implementation of scalar multiplication, mitigating side channel vulnerabilities that are more readily found with other variants of elliptic curves (i.e. the NIST curves) that suffer poor implementations. Past this, every element of curve25519 is well-explained, unlike the NIST curves, which motivates more trust in it. Of additional note, there is a birational equivalence between curve25519's Montgomery form and an Edwards form that is typically used in digital signatures - this means we can do computation on the Montgomery variant and translate back to the Edwards variant for applications where it matters. Here, however, we are only concerned with the Montgomery form and how an unsafe mechanism can still cause issues, despite a safe curve being used.

We will now describe the methodology of the DUAL_EC_DRBG generator, (henceforth referred to as ECDRBG). ECDRBG has three security assertions:

- Elliptic Curve Decisional Diffie-Hellman Problem (ECDDH): Let $\mathcal{E}$ be an elliptic curve defined over a finite field of prime characteristic $\mathbb{F}_q$. Let $G \in \mathcal{E}(\mathbb{F}_q)$ be a generator for an additive cyclic subgroup $\mathcal{E}_\mathcal{G}$ of prime order p over the points of $\mathcal{E}$. Given $G, s_1G, s_2G, Q : s_1, s_2 \in \mathbb{Z}_q$ and s_1, s_2 chosen uniformly at random, decide if $Q = s_1 s_2 G$ or if Q is a random valid point in $\mathcal{E}_G$. This is presumed hard.
- X-Logarithm Problem (X-Log Problem/XLP): Given an elliptic curve point $Q = kP$, determine if k is congruent to the x-coordinate of an elliptic curve point. This problem is questionable, but presumed hard.
- Truncated Point Problem (TPP): Given a bitstring of a specific length n, determine whether it is obtained by truncating the x-coordinate of a random elliptic curve point. This problem is presumed hard when enough bits are truncated.

We will now describe the algorithm. We are given an elliptic curve $\mathcal{E}$ with a finite field of prime characteristic q, and two defined base points $P, Q \in \mathcal{E}(\mathbb{F}_q)$. We are also offered four functions:

- $X(x, y) \rightarrow x$: Extract the x-coordinate of a given elliptic curve point in affine coordinates.
- $t(x) : \{0, 1\}^k \rightarrow \{0, 1\}^{k-n}$: Truncates n MSB of data, with default being 16.
- $g_P(x) = X(xP)$: Performs point doubling of P x times (scalar multiplication).
- $g_Q(x) = t(X(xQ))$: Performs point doubling of Q x times, extracts the affine x-coordinate, and truncates it.

The generator is then very simple. Select a seed $s \in \mathbb{F}_q$, and call $g_P(s)$ to establish the first state V. To generate output, call $g_Q(V)$. V_2 is generated by calling $g_P(V)$. For a true implementation of ECDRBG, one must use the definitions given by NIST as well as a NIST curve. However, for our challenge, we are only interested in ability to implement and utilize the backdoor. As such, truncation is omitted (we want to know if you can implement this and use the backdoor, not if you can rip a brute force), and we do not care about low base-point order attacks since we have defined P, Q in the cyclic subgroup and not in the -torsion group. Hence, 252 bit numbers (with the leading bit set to comply with Montgomery ladder requirements) are used, as defined in the challenge parameters.

We will now go over the mechanism of the backdoor, and the intended path to solve utilizing it. With regards to the NIST curves, we have it that the cofactor is 1, ergo there must necessarily be some scalar $s : sQ = P$. Since this is not the case with curve25519 (although the given P, Q are within the same cyclic subgroup for the challenge), the curious competitor could confirm this quickly by computing a pairing and ensuring it is trivial. Given that it is, we are equipped with a similar scenario: there must be some scalar s such that $sQ = P$. Particularly concerning is that there is no explanation for selection of P, Q, ergo this relationship may be known by the constructing party. Now, given a single output of the generator t, we consider the point A with x-coordinate t. There must, in turn, be some scalar $r : A = rQ$, since outputs are some scalar multiplication of Q. Given a party knows s and A, we demonstrate the backdoor:

- $sA = srQ$ $(A = rQ)$
- $srQ = rP$ $(sQ = P)$
- $rP = state_2$

With a single output, the state can be inferred and the output stream is compromised. Back to the challenge, AES-256-GCM works with both a key and a nonce/IV (we will use nonce moving forward). Given the nonce generated from this generator, one can use the backdoor to deduce the key. Once obtained, simply decrypt the given ciphertext with the valid nonce/key pair and the flag is obtained.

As an aside, this was meant to be a very punishing challenge time-wise - the first three were meant to be quickly solved (although this wasn't actually the case), and the majority of player time was intended to be spent here and on challenge 5.

Challenge 5: Move Over

This is the last challenge in the track. The player is given an elliptic curve, the prime field characteristic p, a point P, a point Q, the prime order r of said points, the Tate pairing $e(kP, Q)$, and an ambiguity factor. They are then tasked to find the value of k, which is the flag.

The crux of this challenge is the Menezes-Okamoto-Vanstone attack, or MOV attack. The general concept is the translation of the Elliptic Curve Discrete Logarithm Problem to the Discrete Logarithm Problem over the integers, which is far more tractable utilizing the index calculus. While the methodology of the index calculus is outside the scope of this document, both a mathematical explanation and link to a state of the art implementation will be provided in the resources section. This is intended to be the most difficult challenge presented in the track. Pairing-based cryptography was selected because we felt that, given the time constraints imposed by Challenge 4, it would be the most naturally extensible and not require as much research as some of the other options that were considered (The Algebraic Eraser and Supersingular Isogeny Diffie-Hellman were also potential final challenges).

The player is first meant to deduce the embedding degree of the elliptic curve, defined as the smallest value $k \in \mathbb{Z} : r \mid p^k - 1$. They should quite quickly find that $k = 1$, indicating the viability of the MOV attack. The curve was generated utilizing a method from "Constructing Pairing-Friendly Curves under Embedding Degree 1 for Securing Critical Infrastructure", which will be offered in the Further Reading section for those interested. An alternative for construction of arbitrary embedding degree pairing friendly curves is the Cocks-Pinch method, to which a reference will also be given.

Past this, we must discuss the meaning of a pairing in the context of elliptic curve cryptography. A bilinear pairing is a function $e : G_1 \times G_2 \to G_T$, where two elements of different groups can be mapped to a third under the following axioms:

- Bilinearity: $\forall a, b \in \mathbb{F}_q^*, P \in G_1, Q \in G_2, e(aP, bQ) = e(P, Q)^{ab}$
- Non-degeneracy: $e \neq 1$

In the case of $e = 1$, we further specify this to be the trivial pairing, or when $P, Q \in G_n$. The two most popular choices of pairing are the Weil Pairing and the Tate Pairing, with several variations existing for the latter. Both pairings will output an r-th root of unity, defined as a value $z \in \mathbb{F}_q : z^r \equiv 1 \bmod q^k$. To further elaborate (because it is relevant), the Tate Pairing requires an additional step to do so, however, it is still often preferred in application because of the efficiency it offers. Additionally, the Weil Pairing requires us to find a field containing all points in $\mathcal{E}[r]$, whereas the Tate Pairing only requires us to find the field containing the r-th roots of unity. When $k > 1$, these conditions are identical by Balasubramanian-Koblitz (BK). We will briefly state BK before further diving into the math behind pairings:

- Balasubramanian-Koblitz Theorem (BK): Let $\mathcal{E}$ be an elliptic curve defined over $\mathbb{F}_q$. Suppose $\exists r : r \mid |\mathcal{E}(\mathbb{F}_q)|, r \nmid q-1$. Then, $\mathcal{E}(\mathbb{F}_{q^k})$ contains r^2 points of order r iff $r \mid q^k - 1$.
- The proof is trivial and left to the reader. Or, you can find it in the Further Reading section, courtesy of Ben Lynn.

We will now briefly discuss Weil Pairings and Tate Pairings before contextualizing this in terms of the CTF. The Weil Pairing is a rational function of the form:

$$e(P, Q) = \frac{f_P(\mathcal{A}_Q)}{f_Q(\mathcal{A}_\mathcal{P})}$$

Where the function f_N denotes a another rational function with divisor $r(N) - r(\mathscr{O})$ for a point N, where r is the order of the point. Since $P, Q \in \mathcal{E}[r]$ and the function is non-zero, this provides a degree zero divisor, matching the expectations for a smooth projective curve over a finite field. We additionally note that $\mathcal{A}_N$ is the non-zero divisor $(P) - (\mathscr{O})$. With this in mind, we find $f_P(\mathcal{A}_Q)$ to be expressed as:

$$\frac{f_P(Q)}{f_p(\mathscr{O})}$$

Which is computable with Miller's Algorithm, described below:

Algorithm 2 Miller's Algorithm

```
 1: procedure MILLER($P, Q, r, p$)
 2:     f := 1
 3:     t := P
 4:     bits := bin(r)[1:]
 5:     for all bits $b$ in bits do
 6:         line := computeLine(t, t, Q, p)
 7:         vert := computeVert(t + t, Q, p)
 8:         f := (f * f * line * vert$^{-1}$) mod p
 9:         t := t + t
10:         if bit then
11:             lineon := computeLine(t, P, Q, p)
12:             nextT := t + P
13:             verton = computeVert(nextT, Q, p)
14:             f = (f * lineon * verton$^{-1}$) mod p
15:             t = nextT
16:         end if
17:     end for
18:     return f
19: end procedure
```

The Tate Pairing is an alternative pairing that is much easier computationally, defined as:

$$e(P, Q) = f_P(\mathcal{A}_Q)$$

Where $P \in \mathcal{E}[r], Q \in \mathcal{E}(\mathbb{F}_q)$ It is computed as you would expect. However, as previously discussed, the requirements on the Tate pairing are easier in embedding degree one: the field must only contain the r-th roots of unity, rather than all r-torsion points. This can be validated by testing if $r \mid |\mathcal{E}(\mathbb{F}_{q^k})| - 1$. However, the output of the Tate Pairing as-is is not sufficient for use in cryptography, because it is not exact, rather simply a representative. To remedy this, we perform Tate exponentiation as:

$$e(P, Q) = f_P(\mathcal{A}_Q)^{q^k - 1/r}$$

where k is the embedding degree of the curve. Notably, both of these are defined only up to equivalence of a root of unity with $k = 1$ - there is some ambiguity factor $\alpha \in \mu_r$, where μ_r is the set of r-th roots of unity. This is why players are provided with an ambiguity factor at the start, as determining what the ambiguity is is a challenge in and of itself.

With the mathematical preliminaries out of the way, we will now MOVe into the MOV attack itself. Given points P, Q, knowledge of the order of P, r, the pairing of $e(kP, Q)$, and the ambiguity factor α, players are tasked to recover k. The most direct answer to this is to utilize a generic DLP solver, say Baby Step Giant Step or Pollard's Rho, or potentially something like Pohlig-Hellman. However, by determining the embedding degree of the curve $k = 1$, this is a clear indication that MOV is much more efficient (indeed, it allows for a solve in about 5 minutes with cado-nfs), whereas the other algorithms will require several hours of computation. Players must compute the Tate Pairing $e(P, Q)$, from which they have derived a base for the DLP given:

$$e(kP, Q) = e(P, Q)^k$$

From this point, players are expected to use cado-nfs (or another optimized index calculus implementation) to compute the discrete logarithm. Cado-nfs has a peculiarity of disallowing a choice of base, and as such, the outputs must be normalized against one another. This is done by computing the target $log_1 = e(kP, Q)$ pairing, then the base $log_2 = e(P, Q)$ pairing, and taking $k_2 = log_1 * log_2^{-1} \bmod r$. Then, by multiplying $\alpha * k_2$, the player should arrive at the secret k, which is the flag.

Concluding Remarks

To those who played in the CTF this year, thank you for taking part, and I hope that both playing and reading this was a fun (of the type II variety, more than likely) and educational experience. For those reading out of curiosity, I hope you play the Cube next year and that this was still able to provide some form of value. Of course, to the winners, congratulations! I hope those black badges bring you back for many years to come. And, of course, to everyone else who was involved in the creation of this year's Cube, thank you for making awesome challenges and piecing everything together into something coherent and interesting. See everyone again next year!

Further Reading

For those interested in the topics discussed here, more in-depth papers, resources, and books are offered below. They are hyperlinked when available for your convenience.

- Wikipedia - Linear Congruential Generators
- Haldir - How to Crack a Linear Congruential Generator
- Hanno Böck - Fermat Factorization
- Abhay Chennagiri - A Tutorial Paper on Håstad's Broadcast Attack
- University of Maryland - RSA Attacks
- Ben Lynn - The Chinese Remainder Theorem
- Ferguson, Shumow - On The Possibility of a Back Door in the NIST SP800-90 DUAL_EC_PRNG
- Bernstein, Lange, Niederhagen - DUAL EC: A Standardized Back Door
- Washington - Elliptic Curves: Number Theory and Cryptography
- MIT OCW - 18.783 Elliptic Curves
- Ben Lynn - Elliptic Curves

- Wang, Dai, Choo, Jayaraman, Ranjan - Constructing Pairing-Friendly Elliptic Curves Under Embedding Degree 1 for Securing Critical Infrastructure
- Ben Lynn - Bilinear Pairings
- Ben Lynn - Balasubramanian-Koblitz Theorem
- Ben Lynn - The Tate Pairing
- Katz, Lindell - Introduction to Modern Cryptography
- 1nfocalypse - Cracking Chaos: Making, Using, and Breaking PRNGs