Consider the elliptic curve $E$ over $\mathbb{F}_{2^k}$, where $|E| = n$.

Assume we want to solve the elliptic curve discrete logarithm problem: find $k$ in $Q = kP$. 

Aaron Blumenfeld

Pollard’s Rho Algorithm
Pollard’s Rho Algorithm

Partition $E$ into $S_1 \cup S_2 \cup S_3$, where the $S_i$ are similar in size.

Choose $A_i \in E$ as some scalar multiple of $P$.

Let $A_{i+1} = f(A_i) = \begin{cases} 
A_i + P, & A_i \in S_1, \\
2A_i, & A_i \in S_2, \\
A_i + Q, & A_i \in S_3.
\end{cases}$
Pollard’s Rho Algorithm

Image credit: Washington [1]
The terms of the sequence then take the form $A_i = a_j P + b_j Q$.

Once we see an equality $A_{i_1} = A_{i_2}$, we have

$$a_{j_1} P + b_{j_1} Q = a_{j_2} P + b_{j_2} Q,$$

which means that

$$\frac{a_{j_1} - a_{j_2}}{b_{j_2} - b_{j_1}} P = Q.$$

The ECDLP can thus be solved provided that $\gcd(b_{j_2} - b_{j_1}, n) = 1$. 
In fact, even if \( \gcd(b_{j_2} - b_{j_1}, n) = d > 1 \), we can compute

\[
\frac{a_{j_1} - a_{j_2}}{b_{j_2} - b_{j_1}} \pmod{N/d}.
\]

There are then \( d \) possibilities for \( k \), which is only intractable for large \( d \).

In practice, however, \( d \) is quite small, especially if \( E \) is chosen so that \( n \) is prime.
Unlike Baby-Step Giant-Step, only $O(1)$ space complexity is required:

Start with the ordered pair $(A_1, A_2)$. Given $(A_i, A_{2i})$, we can compute $(A_{i+1}, A_{2i+2}) = (f(A_i), f(f(A_{2i})))$. 
Why does this find a match?

- Suppose \( A_i = A_j \). Then \( A_{i+k} = A_{j+k} \) for all \( k \geq 0 \).
- For \( k = j - 2i (\geq 0) \), we have \( A_{i+j-2i} = A_{j+j-2i} \), or \( A_{j-i} = A_{2(j-i)} \).
- Note that \( j - i \geq i \) by construction since \( j \geq 2i \).
However, it turns out that this function $f$ performs approximately 33% more slowly than the expectation.

It can be shown that the tail and cycle length both have an expectation of $\sqrt{\pi n/8}$.

Therefore, a cycle should be detected within $2\sqrt{\pi n/8} = \sqrt{\pi n/2}$ iterations.
Increasing Number of Partition Elements

- Research has indicated that using more than 3 partition elements improves the randomness of the function $f$.
- This improves the performance of the algorithm.
In order to do this, we can hash the points \((x, y) \in E\) to the set \(\{1, \ldots, m\}\).

- It turns out hashing based on the \(x\)-coordinate is just as effective as using the \(y\)-coordinate.
- Since the \(x\)-coordinate is a polynomial, we can represent it as a binary vector and view it as an integer for the purposes of hashing.
- We then partition evenly into \(m\) subsets of size \(\frac{2^k}{m}\).
We define $M_j = a_jP + b_jQ$, where the $a'_j$s and $b'_j$s are randomly chosen modulo $n$.

We then define $f(A_i) = A_i + M_j$ when $A_i \in S_j$. 

The best choice for $m$ in simulating a random function $f$ seems to be in the range [20, 30].

However, there is evidence that for $m$ around 60, the function $f$ performs more efficiently than a random map by about 6%.
Future Work

- Collect statistics for curves over larger binary fields (the data gathered was for curves over $\mathbb{F}_{2^8}$).
- Perform similar analysis for curves over $\mathbb{F}_p$. 
