International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Volume 4, Issue 3 (May-June, 2016), PP. 189-193

METHOD TO DECRYPT AN ENCRYPTED SCRIPT USING NEWTON DIFFERENCE FORMULA FOR UNEQUAL SPACING OF INTERVAL

Madhusudan Verma
Vellore Institute of Technology, Chennai , Tamilnadu ,India.

Abstract— **Cybersecurity plays an important role in day to day life. Hackers try to decrypt the code. The hashing function is defined in such a way so that it cannot be inverted. Various methods to decrypt has been developed but there are some encryptions which are difficult to decrypt. These paper discusses about decrypting code which has been encrypted using hash function using newton difference formula for unequal spacing**

Index terms- **hashing, prehashing, newton difference formula, cybersecurity.**

I. INTRODUCTION

Cryptography is a powerful to protect information for computer system. Various

encryption methods are hashing, symmetric methods and asymmetric methods.

Hashing: Before hashing prehashing is done in which any data type is converted to integer.Hashing is a mapping from key to value. Hashing creates a unique, fixed-length signature for a message or data set. Each "hash" is unique to a specific message, so minor changes to that message would be easy to track. Once data is encrypted using hashing, it is difficult to be reversed or deciphered.

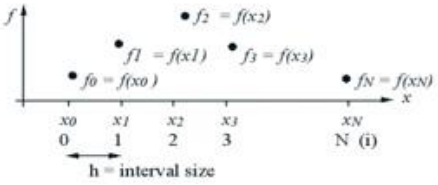
Symmetric encryption: Symmetric encryption's job is to take readable data ("plaintext"

in crypto parlance), scramble it to make it unreadable (protecting it from prying eyes while it's being stored on a disk or transmitted over a network), then unscramble it again when it's needed. It's generally fast, and there are lots of good encryption methods to choose from. The most important thing to remember about symmetric encryption is that both sides—the encrypter, and the decrypter—need access to the same key.

Asymmetric encryption: Asymmetric encryption also takes readable data, scrambles it,and unscrambles it again at the other end, but there's a twist: a different key is used foreach end. Encrypters use a public key to scramble the data, and decrypters use the matching private (secret) key on the other end to unscramble it again.The purpose of this paper to find the key for the values of inverse hash function.

Newton Forward Difference method

• We assume equi-spaced points (not necessary)



• Forward differences are now defined as follows:

$\Delta^0 f_i = f_i$ (Zeroth order forward difference)

$\Delta f_i = f_{i+1} - f_i$ (First order forward difference)

$\Delta^2 f_i = \Delta f_{i+1} - \Delta f_i$ (Second order forward difference)

$\Delta^2 f_i = (f_{i+2} - f_{i+1}) - (f_{i+1} - f_i)$

$\Delta^2 f_i = f_{i+2} - 2f_{i+1} + f_i$

$\Delta^3 f_i = \Delta^2 f_{i+1} - \Delta^2 f_i$ (Third order forward difference)

$\Delta^3 f_i = (f_{i+3} - 2f_{i+2} + f_{i+1}) - (f_{i+2} - 2f_{i+1} + f_i)$

$\Delta^3 f_i = f_{i+3} - 3f_{i+2} + 3f_{i+1} - f_i$

$\Delta^k f_i = \Delta^{k-1} f_{i+1} - \Delta^{k-1} f_i$ (k^{th} order forward difference)

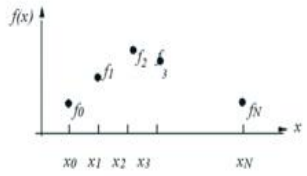
• Typically we set up a difference table

i	f_i	Δf_i	$\Delta^2 f_i$	$\Delta^3 f_i$	$\Delta^4 f_i$
0	f_o	$\Delta f_o = f_1 - f_o$	$\Delta^2 f_o = \Delta f_1 - \Delta f_o$	$\Delta^3 f_o = \Delta^2 f_1 - \Delta^2 f_o$	$\Delta^4 f_o = \Delta^3 f_1 - \Delta^3 f_o$
1	f_1	$\Delta f_1 = f_2 - f_1$	$\Delta^2 f_1 = \Delta f_2 - \Delta f_1$	$\Delta^3 f_1 = \Delta^2 f_2 - \Delta^2 f_1$	
2	f_2	$\Delta f_2 = f_3 - f_2$	$\Delta^2 f_2 = \Delta f_3 - \Delta f_2$		
3	f_3	$\Delta f_3 = f_4 - f_3$			
4	f_4				

To compute higher order differences in the tables, we take forward differences of previous order differences instead of using expanded formulae.

. N^{th} order forward differences requires N+1 data points.

Overview of Derivation of Newton Forward Interpolation on Equi-spaced Points



- Summary of Steps
  - Step 1: Develop a general Taylor series expansion for $f(x)$ about x_o .
  - Step 2: Express the various order forward differences at x_o in terms of $f(x)$ and its derivatives evaluated at x_o . This will allow us to express the actual derivatives eval-uated at x_o in terms of forward differences.
  - Step 3: Using the general Taylor series expansion developed in Step 1, sequentially substitute in for the derivatives evaluated at x_o in terms of forward differences (i.e. substitute in the expressions developed in Step 2).

Step 1

- The Taylor series expansion for $f(x)$ about x_o is:

$$f(x) = f(x_o) + (x-x_o)\frac{df}{dx}\Big|_{x=x_o} + \frac{1}{2!}(x-x_o)^2\frac{d^2f}{dx^2}\Big|_{x=x_o} + \frac{1}{3!}(x-x_o)^3\frac{d^3f}{dx^3}\Big|_{x=x_o} + O(x-x_o)^4$$

$$\Rightarrow$$

$$f(x) = f_o + (x-x_o)f_o^{(1)} + \frac{1}{2!}(x-x_o)^2 f_o^{(2)} + \frac{1}{3!}(x-x_o)^3 f_o^{(3)} + O(x-x_o)^4$$

Step 2a

- Express first order forward difference in terms of f_o , $f_o^{(1)}, \ldots$

$$\Delta f_o \equiv f_1 - f_o$$

- However since $f_1 = f(x_1)$, we can use the Taylor series given in *Step 1* to express f_1 in terms of f_o and its derivatives:

$$f_1 = f_o + (x_1-x_o)f_o^{(1)} + \frac{1}{2!}(x_1-x_o)^2 f_o^{(2)} + \frac{1}{3!}(x_1-x_o)^3 f_o^{(3)} + O(x_1-x_o)^4$$

Step 2b

- Express second order forward difference in terms of f_o, $f_o^{(1)}, \ldots$

$$\Delta^2 f_o \equiv f_2 - 2f_1 + f_o$$

- We note that $f_1 = f(x_1)$ was developed in T.S. form in *Step 2a.*
- For $f_2 = f(x_2)$ we use the T.S. given in *Step 1* to express f_2 in terms of f_o and derivatives of f evaluated at x_o

$$f_2 = f_o + (x_2-x_o)f_o^{(1)} + \frac{1}{2!}(x_2-x_o)^2 f_o^{(2)} + \frac{1}{3!}(x_2-x_o)^3 f_o^{(3)} + O(x_2-x_o)^4$$

- We note that $x_2 - x_o = 2h$

$$f_2 = f_o + 2hf_o^{(1)} + \frac{4}{2!}h^2 f_o^{(2)} + \frac{8}{3!}h^3 f_o^{(3)} + O(h)^4$$

Step 2c

- Express the third order forward difference in terms of f_o, $f_o^{(1)} \ldots$

$$\Delta^3 f_o \equiv f_3 - 3f_2 + 3f_1 - f_o$$

- We already developed expressions for f_2 and f_1.
- Develop an expression for $f_3 = f(x_3)$ using the T.S. in *Step 1*

$$f_3 = f_o + (x_3-x_o)f_o^{(1)} + \frac{1}{2!}(x_3-x_o)^2 f_o^{(2)} + \frac{1}{3!}(x_3-x_o)^3 f_o^{(3)} + O(x_3-x_o)^4$$

- Noting that $x_3 - x_o = 3h$

$$f_3 = f_o + 3hf_o^{(1)} + \frac{9}{2}h^2 f_o^{(2)} + \frac{9}{2}h^3 f_o^{(3)} + O(h)^4$$

- Now substitute in for f_2 and f_1 into the definition of the second order forward difference operator

$$\Delta^2 f_o = f_o + 2hf_o^{(1)} + 2h^2 f_o^{(2)} + \frac{4}{3}h^3 f_o^{(3)} + O(h)^4 \quad -2f_o - 2hf_o^{(1)} - h^2 f_o^{(2)} - \frac{1}{3}h^3 f_o^{(3)} + O(h)^4 + f_o$$

$$\Rightarrow$$

$$f_o^{(2)} = \frac{\Delta^2 f_o}{h^2} - hf_o^{(3)} + O(h)^2$$

- Note that the second order forward difference divided by h^2 is in fact an approximation to $f_o^{(2)}$ to $O(h)$. However, we will use all terms in the expression.

- We note that the spacing between data points is $h \equiv x_1 - x_o$:

$$f_1 = f_o + hf_o^{(1)} + \frac{1}{2!}h^2 f_o^{(2)} + \frac{1}{3!}h^3 f_o^{(3)} + O(h)^4$$

- Now, substitute in for f_1 into the definition of the first order forward differences

$$\Delta f_o = f_o + hf_o^{(1)} + \frac{1}{2!}h^2 f_o^{(2)} + \frac{1}{3!}h^3 f_o^{(3)} + O(h)^4 - f_o$$

$$\Rightarrow$$

$$f_o^{(1)} = \frac{\Delta f_o}{h} - \frac{1}{2!}hf_o^{(2)} - \frac{1}{3!}h^2 f_o^{(3)} - O(h)^3$$

- Note that the first order forward difference divided by h is in fact an approximation to the first derivative to $O(h)$. However, we will use all the terms given in this sequence.

- Substituting in for f_3, f_2 and f_1 into the definition of the third order forward difference formula.

$$\Delta^3 f_o = f_o + 3hf_o^{(1)} + \frac{9}{2}h^2 f_o^{(2)} + \frac{9}{2}h^3 f_o^{(3)} + O(h)^4 \quad -3f_o - 6hf_o^{(1)} - \frac{12}{2}h^2 f_o^{(2)} - \frac{24}{3!}h^3 f_o^{(3)} + O(h)^4$$

$$+3f_o + 3hf_o^{(1)} + \frac{3}{2}h^2 f_o^{(2)} + \frac{3}{3!}h^3 f_o^{(3)} + O(h)^4 - f_o$$

$$\Rightarrow$$

$$\Delta^3 f_o = h^3 f_o^{(3)} + O(h)^4$$

$$\Rightarrow$$

$$f_o^{(3)} = \frac{\Delta^3 f_o}{h^3} + O(h)$$

- The third order forward difference divided by h^3 is an $O(h)$ approximation to $f_o^{(3)}$

Step 3a

- Consider the general T.S. expansion presented in *Step 1* to define $f(x)$ and substitute in for $f_o^{(1)}$ using the result in *Step 2a*.

- Note that now we are ***not*** evaluating the T.S. at a data point but at any x

$$f(x) = f_o + (x - x_o)\left[\frac{\Delta f_o}{h} - \frac{1}{2!}hf_o^{(2)} - \frac{1}{3!}h^2 f_o^{(3)} - O(h)^3\right]$$
$$+\frac{1}{2!}(x - x_o)^2 f_o^{(2)} + \frac{1}{3!}(x - x_o)^3 f_o^{(3)} + O(x - x_o)^4$$

$$\Rightarrow$$

$$f(x) = f_o + \frac{x - x_o}{h}\Delta f_o + \frac{1}{2!}[-(x - x_o)h + (x - x_o)^2]f_o^{(2)}$$
$$+\frac{1}{3!}[-(x - x_o)h^2 + (x - x_o)^3]f_o^{(3)} + O(h)^4$$

Step 3b

- Substitute in for $f_o^{(2)}$ using the expression developed in *Step 2b*.

$$f(x) = f_o + \frac{x - x_o}{h}\Delta f_o + \frac{1}{2!}[-(x - x_o)h + (x - x_o)^2]\left[\frac{\Delta^2 f_o}{h^2} - hf_o^{(3)} + O(h)^2\right]$$
$$+\frac{1}{3!}[-(x - x_o)h^2 + (x - x_o)^3]f_o^{(3)} + O(h)^4$$

$$\Rightarrow$$

$$f(x) = f_o + \frac{x - x_o}{h}\Delta f_o + \frac{1}{2!}\left[-\frac{(x - x_o)}{h} + \frac{(x - x_o)^2}{h^2}\right]\Delta^2 f_o$$
$$+\frac{1}{3!}[2(x - x_o)h^2 + (x - x_o)^3 - 3(x - x_o)^2 h]f_o^{(3)} + O(h)^4$$

Step 3c

- Substitute in for $f_o^{(3)}$ from *Step 2c*

$$f(x) = f_o + \frac{(x - x_o)}{h}\Delta f_o + \frac{1}{2!}\left[-\frac{(x - x_o)}{h} + \frac{(x - x_o)^2}{h^2}\right]\Delta^2 f_o$$
$$+\frac{1}{3!}[2(x - x_o)h^2 + (x - x_o)^3 - 3(x - x_o)^2 h]\frac{\Delta^3 f_o}{h^3} + O(h)^4$$

- Re-arranging the terms in brackets:

$$f(x) = f_o + (x - x_o)\frac{\Delta f_o}{h} + \frac{1}{2!}(x - x_o)[-h + (x - x_o)]\frac{\Delta^2 f_o}{h^2}$$
$$+\frac{1}{3!}(x - x_o)[2h^2 + (x - x_o)^2 - 3(x - x_o)h]\frac{\Delta^3 f_o}{h^3} + O(h)^4 + HOT$$

$$f(x) = f_o + (x - x_o)\frac{\Delta f_o}{h} + \frac{1}{2!}(x - x_o)[x - (x_o + h)]\frac{\Delta^2 f_o}{h^2}$$
$$+\frac{1}{3!}(x - x_o)[(x - (x_o + h))(x - (x_o + 2h))]\frac{\Delta^3 f_o}{h^3} + O(h)^4 + HOT$$

- Also considering higher order terms ***and*** noting that $x_o + h = x_1$, $x_o + 2h = x_2$ and $f(x) = g(x) + e(x)$

$$g(x) = f_o + (x - x_o)\frac{\Delta f_o}{h} + \frac{1}{2!}(x - x_o)(x - x_1)\frac{\Delta^2 f_o}{h^2} + \frac{1}{3!}(x - x_o)(x - x_1)(x - x_2)\frac{\Delta^3 f_o}{h^3}$$
$$+ \ldots + \frac{1}{N!}(x - x_o)(x - x_1)(x - x_2)\ldots(x - x_{N-1})\frac{\Delta^N f_o}{h^N}$$

- **This is the N^{th} degree polynomial approximation to $N+1$ data points and is identical to that derived for Lagrange interpolation or Power series (only the form in which it is presented is different).**

- Note that the $N+1$ data point are ***exactly*** fit by $g(x)$

$$g(x_o) = f_o$$

$$g(x_1) = f_o + (x_1 - x_o)\frac{f_1 - f_o}{h} = f_o + h\left(\frac{f_1 - f_o}{h}\right) = f_1$$

$$g(x_2) = f_o + (x_2 - x_o)\left(\frac{f_1 - f_o}{h}\right) + \frac{1}{2}(x_2 - x_o)(x_2 - x_1)\frac{1}{h^2}(f_2 - 2f_1 + f_o)$$

$$\Rightarrow$$

$$g(x_2) = f_o + \frac{2h}{h}(f_1 - f_o) + \frac{(2h)h}{2h^2}(f_2 - 2f_1 + f_o)$$

$$\Rightarrow$$

$$g(x_2) = f_o + 2f_1 - 2f_o + f_2 - 2f_1 + f_o = f_2$$

- ***In general***

$$g(x_i) = f_i \qquad i = 0, N$$

- It can be readily shown that the error at any x is: (by carrying through error terms in the T.S.)

$$e(x) = f(x) - g(x) = \frac{(x - x_o)(x - x_1)\ldots(x - x_N)}{(N+1)!} f^{(N+1)}(\xi) \qquad x_o < \xi < x_N$$

- This error function is identical to that for Lagrange Interpolation (since the polynomial approximation is the same).

- However we note that $f^{N+1}(x)$ can be approximated as (can be shown by T.S.)

$$f^{(N+1)}(x_o) \cong \frac{\Delta^{N+1} f_o}{h^{N+1}}$$

- In fact if $f^{(N+1)}(x)$ does not vary dramatically over the interval

$$f^{(N+1)}(\xi) \cong \frac{\Delta^{N+1} f_o}{h^{N+1}}$$

Newton Backward Interpolation

- Newton backward interpolation is essentially the same as Newton forward interpolation except that backward differences are used

- Backward differences are defined as:

$\nabla^0 f_i \equiv f_i$ Zeroth order backward difference

$\nabla f_i = f_i - f_{i-1}$ First order backward difference

$\nabla^2 f_i = \nabla f_i - \nabla f_{i-1}$ Second order backward difference

$\nabla^k f_i = \nabla^{k-1} f_i - \nabla^{k-1} f_{i-1}$ k^{th} order backward difference

- For $N+1$ data point which are fitted with an N^{th} degree polynomial

$$g(x) = f_N + (x - x_N)\frac{\nabla f_N}{h} + \frac{1}{2!}(x - x_N)(x - x_{N-1})\frac{\nabla^2 f_N}{h^2}$$

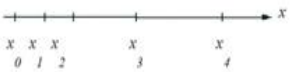
$$+ \frac{1}{3!}(x - x_N)(x - x_{N-1})(x - x_{N-2})\frac{\nabla^3 f_N}{h^3}$$

$$+ \frac{1}{N!}(x - x_N)(x - x_{N-1})\ldots(x - x_1)\frac{\nabla^N f_N}{h^N}$$

- Note that we are really expanding about the right most point to the left. Therefore we must develop f_N, ∇f_N etc. in the difference table

Newton Interpolation on Non-uniformly Spaced Data Points

- Newton interpolation can be extended for non-uniformly spaced data points



- A difference table for non-uniformly points is developed and an appropriate interpolation formula is developed and used .

Main idea for decryption

The first step is to covert encrypted script to a integers. As keys are converted to numbers already during prehashing it is not required to convert into integer. Assume a discrete inverse hashing function as a continuous function. The purpose for doing so is so that we can find a polynomial interpolating function by using Newton difference formula for unequal spacing. We require unequal spacing because we have values which can be mapped to any key.

Since we do not know inverse hash function create a table which have N+1 data points. Find Nth degree polynomial interpolation function using N+1 data points and Newton forward or backward difference formulae . If the value of the independent variable is closer to the starting numbers of table newton forward difference formula is used

otherwise if the value of the independent variable is closer to the ending numbers of table newton backward difference formula is used.

II. Implementation:

A code in c has been written and tested that actually found the keys for values in mapping without knowing the inverse hash function.

III. Conclusion

We have seen various methods of encryption and seen interpolation of a function using Newton difference formula can be used as an effective tool to find the approximate values of a function at any point. We have also seen how by finding approximation value of **unknown** inverse hash function a text can be decrypted which is encrypted using hash function. So, this method can be used as a powerful tool in cryptanalysis.

REFERENCES

https://www3.nd.edu/~coast/jjwteach/www/www/.../lecture 4_7v14.pdf

http://www.wisegeek.org/what-are- the-different- types-of-encryption-methods.htm#didyouknowout

http://datashieldcorp.com/2013/06/04/3-different-data-encryption-methods/

http://homepages.uel.ac.uk/u0430614/Encryption%20index .htm

http://www.kmitldss.org/kmitldss/articles/fde_p3.pdf

http://nptel.ac.in/courses/122104018/node109.html

https://mat.iitm.ac.in/home/sryedida/public_html/caimna/interpolation/nfdf.html

http://www.sakshieducation.com/Engg/EnggAcademia/CommonSubjects/MathMethods-Interpolation.pdf