

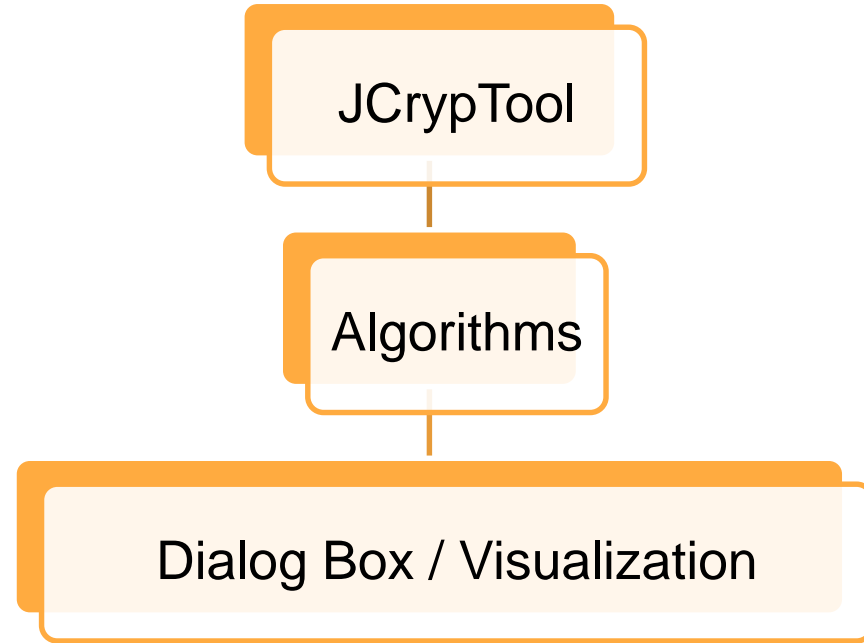
Cryptology

Dec 12 2016

[GitHub](#)

[Trello](#)

System Diagram



Product Highlights

1. Provide users with SHA3 candidates to choose when using JCrypTool.
2. Design a better dialog box for users to choose different algorithms and read the output more easily.
3. Visualize the Blake algorithm.

Demonstration

Test Results

Problem with Algorithms

The algorithms didn't output the correct answer.

Test case(for Groestl):

1. Abcd: D8CD6BE396A8F029BC46E48367D3D84150776C10B7A6AFFEDB19E8D0D175A708
2. Aefg: D8CD6BE396A8F029BC46E48367D3D84150776C10B7A6AFFEDB19E8D0D175A708

Problem 1: Unless we change the first character of the input, the output will be the same.

Problem 2: Unless we change the length of the input, the output will be the same for most cases.

Problem solved:

1. Abcd: 4A639E2F274A8B6A4D3AC957456F8FAE8DF80E2AEA89C19276BDACA5586B5F99
2. Aefg: EDFF182DB0D66874D155CA79CD37FF3F625C822B37AE66E3C1903D50AB5C6C3A

Problem with Visualization

Visualization cannot jump automatically when clicking at the button.

Solved: Inp

Also add a

The screenshot shows the JCryptTool interface for SHA3 Candidates. The 'Blake2b' hash function is selected. The input string is 'fqrq3ve'. The salt is 'uu'. A red box highlights the error message 'salt must be digit!!'. The 'Message' and 'Salt' buttons are visible. The 'Inside Hash Function' section shows the initialization process, including the message and salt being padded to a length of 512 bits. The visualization shows the message and salt being padded to a length of 512 bits, and the resulting hash value is displayed.

SHA3 Candidates
A hash function maps a large input set to a smaller output set. The input is of variable length, however, the output usually has a small.

Hash Input
Select a hash function
BLAKE (224 bits)

Representation of the hash values
☒ hexadecimal ☐ decimal ☐ binary

Input String
fqrq3ve

Salt
uu

salt must be digit!!

Hash value
40 E4 BB E0 7A 19 CE 9B AC B1 90 D5 24 E8 41 99 21 82

Hash value (salt added)
40 E4 BB E0 7A 19 CE 9B AC B1 90 D5 24 E8 41 99 21 82 73 DB D0 4F 85 3F EA 93 42 58

Inside Hash Function
Initialization Compressing Sigma Output

BLAKE224/256
First the message is extended so that its length is congruent to 447 modulo 512.
Length extension is performed by appending a bit 1 followed by a sufficient number of 0 bits.
At least one bit and at most 512 are appended.
Then a bit 1 is added, followed by a 64-bit unsigned big-endian representation of . Padding
can be represented as m0...m, 1000...0001, <64>.
This procedure guarantees that the bit length of the padded message is a multiple of 512.

Message

Salt

Initialization

Compressing

Sigma

Output

m0 m1 m2 m3
m4 m5 m6 m7
m8 m9 m10 m11
m12 m13 m14 m15

v0 v1 v2 v3
h0 h1 h2 h3
v4 v5 v6 v7
h4 h5 h6 h7
v8 v9 v10 v11
s0 XOR c0 s1 XOR c1 s2 XOR c2 s3 XOR c3
v12 v13 v14 v15
t0 XOR c4 t0 XOR c5 t1 XOR c6 t1 XOR c7

v0 v1 v2 v3
v4 v5 v6 v7
v8 v9 v10 v11
v12 v13 v14 v15

Test Case 1

1. Different input with same length with 256 hash bit length.

Input1: ABCDefgh

Input2: ABCDijkl

Groestl:

Output1:

AA7FF5E8167C47B451C75852A2CF77362708F9D9C15CE349E26D81FB8266B085

Output2:

CE4D4AE59B17F7441F638E519D8A24BE737CDAF9E6A944BF7BF4A609E184A7BF

Blake:

Output1:

E6A2B048BAB77FC6798346A34FE91635507DFABF21CF904D63DD12933675C6DB

Output2:

0FFFB62363102F94C5FA4791F565CA93FAD8BE51EC2067E16F2BED91310E8C7B

Keccak:

Output1:

4A441AB0A95C5CF500363FA8753AE8E4FC5FB5E3EC54BA492F52271835BBC718

Output2:

0235DC28399BF9B391831E2ABDDC156D1F3A920C9FFE3C4ED1B5ABCBC9DFE1EA

Test Case 2

2. Different input with only ONE character different

Input1: uihkjllihjkl1oiuhjnkml7liu

Input2: uihkjllihjkd1oiuhjnkml7liu

Groestl:

Output1:

EE2BE9B9D85A5F72C11AA258742E1B7E8742C1395966B4CF9D5F66F7780AFA79

Output2:

99FDA91895657FFE1D19F4821945D5750CB9C8F19065F2E413D2EF7A1B904AD4

Blake:

Output1:

C054E0ECCC6AD3DE0E539B9F04904C07B6B75ABF528E7A2644F790C4D67B5447

Output2:

42632FE4B23C9C8DCB2C40503D16674EC315DA0BD5EFB1180A985D86D282DF79

Keccak:

Output1:

F802E85C8EB4E33BD81F2F08CA8A8D93D8F33CC69A93F7AEE9E2395FE15D6217

Output2:

61FF7DD62E23B9B99273ACBEFE04E6456A4D8D2EF2E7783DF093635AA0CB43EE

Top three things that worked well

1. Algorithm implementation for SHA3 candidates.
2. Design the dialog box.
3. Visualize the Blake algorithm.

Top three things that we have learned

1. Learn how to develop a plugin based on a project.
2. Java knowledge, especially the basic Java data structure.
3. Learn how to use Github, Trello and Slack.

Future plan

1. Build more visualization for SHA3 candidates.
2. Implement the visualization by import the package other than have a copy for the visualization
3. Check and package all code together and send it to JCrypTool and push the code to JCrypTool's Github master branch.

The screenshot displays a workspace titled "Cryptology Project" with a sidebar on the left containing navigation links: "To Do", "Doing", "Done", "Research", "Help", "System 4", "System 3", "System 2", "System 1", and "Home". The main area is divided into five columns representing days from Monday to Friday. Each column contains a list of tasks with progress indicators (e.g., "Add a card", "Done", "In Progress").

Monday	Tuesday	Wednesday	Thursday	Friday
<p>To Do</p> <ul style="list-style-type: none"> Add a card 	<p>Doing</p> <ul style="list-style-type: none"> Add a card 	<p>Done</p> <ul style="list-style-type: none"> Add help Text about BLAKE and -convert Research digital signatures based on cryptographic hash functions Create a document to describe all of the world's in the existing cryptopkg package that we will be expected to implement in our algorithm Explore existing SHA3 candidates on JCryptTool software Research Blake2 Cipher Check out Elliptic Development, not Java Check out J_Cryptool software Look into contacting the owners of J_Cryptool 	<p>Help</p> <ul style="list-style-type: none"> Needless Help Agenda 2 Doing Add a card 	<p>System 4</p> <ul style="list-style-type: none"> Finish visualization plug in for other algorithms, General Keccak, CLOC, CLOC2 Finish Keccak algorithm Finish Keccak algorithm Check and package all code together and send it to J_CryptTool and push the code to J_CryptTool Github Run J_CryptTool plug-ins on Blake2 to make sure it adheres to their coding style

Thank You