Cryptal: cryptography more robust than crystal

Carolina Almirola | Jaewan Bahk | Rahul Kapur | Michail Oikonomou | Sammy Tbeile
ca2636 | jb3621 | rk2749 | mo2617 | st2198

1 Language Description

Cryptal is a language designed to handle applications of number theory and encryption. Based on the C language, with syntax clearer and more robust than crystal, it will provide the user with a simple and intuitive environment from which to solve modular arithmetic problems. In addition, it will provide calls for various encryption methods and accept input, such as image files, to be easily and quickly encrypted.

One encryption methods we want to employ is the use of Merkle Trees. Merkle Trees allow for a concise way for us to check the version or potential corruption of our input data as compared to a copy of the original. This data structure underlies the versioning system employed by Git and the consensus mechanism established by cryptocurrencies such as Bitcoin and Ethereum. It is lightweight way to compare potentially large chunks of input data for tampering or corruption.

2 Language Overview

2.1 Motivation

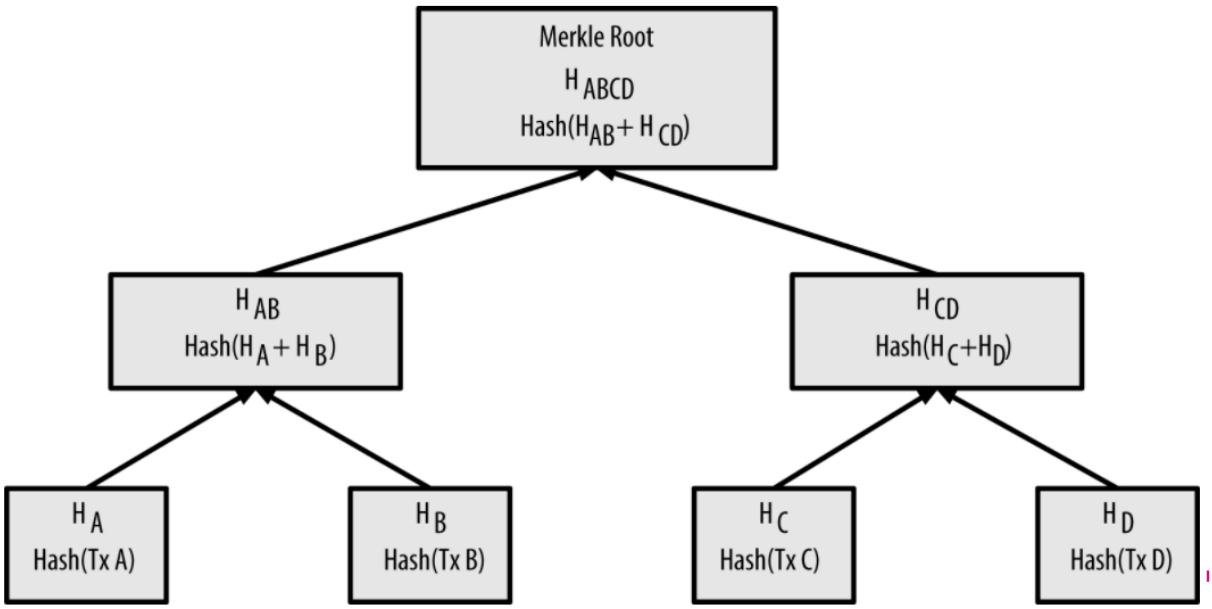
Setting up a modular environment from which to solve modular arithmetic problems and quickly and easily encrypt is non-trivial in most programming languages. We want to facilitate this process by implementing a programming language that focuses in this domain.

2.2 Applications

- Programs that take in files as input and attempt to encrypt/decrypt them
- The Merkle Tree can be used to compare images for pixel level differences or large texts for spacing differences by comparing only comparing 256 bit hashes as opposed to the entire data file
- Programs that make heavy use of modular arithmetic
- Creating fun artwork by encrypting pictures (like the ECB penguin)

3 Features

- Ability to set modulus environment (e.g. calling set mod=x will set the working environment to mod x)
- Calls for popular encryption methods (e.g. RSA, Diffie-Hellman, ElGamal, Elliptic Curves, etc.)
- Ability to perform modular arithmetic when modulus environment is set up.
- Decryption for supported encryption methods.
- Ability to input and process (encrypt) files.



- The Merkle Tree structure contains the input data broken into chunks at the leaves of the tree. Let?s assume a tree of depth d. The parent nodes to the leaves at depth d-1 are formed by finding the hash of two of the leaf nodes. The nodes at depth d-2 are formed by the hash of of two nodes at d-1. And so on, and so forth. Finally, at the root of the tree there is a Merkle Root, which is a hash that represents the entire file and can be used for a quick comparison to see if the input file has been corrupted or not

4 Sample program: encrypting an image

```
import io
import Image

using mod 5

int main(int argc, char** argv){
        if (argc < 3)
                printf("no argument");
        char *filename = argv[1];
        FILE fin = io.open(filename);
        Image orig_im = Image(io.read(fin), Image.jpg);
        Image encrypted_im = Image();

        encrypt orig_im, encrypted_im
        File fout = io.open(filename);
        io.write(fout, Image.toOutStream(encrypted_im, Image.jpg))

        return 0;
}
```