CrackNet (pre Alpha release whitepaper)
Blockchain Based Password Cracking System
August 2019
CrackNet
https://cracknet.io
cracknet@protonmail.com

Abstract

The document details the early stages of CrackNet's development.
CrackNet enables internet users to directly reward the cracking of cryptographic hashes using the Ethereum blockchain to recover passwords or for security research.
Analysis is made of the applicability, effectiveness and risk of CrackNet against different potential targets.
The cryptocurrency token CrackCoin is introduced which is used to store the cryptocurrency reward.
Calculations and predictions are made for the profitability of CrackNet which suggests greater profitability than cryptocurrency mining.
We use some formulas to optimize for maximum profit in CrackNet.

Disclaimer

This document and any information contained may not be sent wholly or in part to any jurisdiction where it would be illegal to buy,sell or trade cryptocurrencies.
No warranty,guarantee or undertaking is made or given regarding

- The accuracy of information contained herein
- The accuracy of any calculations contained herein
- Any financial predictions decribed or calculated

Indemnification

The developer of CrackNet does not encourage, promote ,endorse or recommend computer crime, which may include illegal hacking, fraud, identity theft,carding, theft of computers or mobile devices, leaking or any other computer related activities that may be illegal in your jurisdiction. As such you indemnify the CrackNet developers, promoters and any other involved team members from any responsibility for any crime,criminal damage, damage to health, loss of profits or any other unfortunate circumstance to any individual, corporation, organization, government or any other entity, which may occur as a result of using the knowledge herein.

Definitions

ASCII – American Standard Code for Information Interchange

Attacker – An entity seeking to crack a ciphertext or gain access to a target data of software application

Blockchain - A blockchain, originally block chain, is a continuously growing list of records, called blocks, which are linked and secured using cryptography[1]

Ciphertext – for the purposes of this project this definition shall also include cryptographic hashes

Criticality- The criticality of a ciphertext is whether or not the attacker can afford for anyone else to read the targetdata

Cryptocurrency – A cryptocurrency (or crypto currency) is a designed to work as a medium of exchange that uses strong cryptography to secure financial transactions, control the creation of additional units, and verify the transfer of assets.[2,3,4]

Dedicated Cracking System (DCS) – A computer system that is designed or specifically used to decrypt ciphertext or cryptographic hashes

Hash function – A hash function is any function that can be used to map data of arbitrary size onto data of a fixed size[5] cryptographic hashes are designed to be virtually impossible to calculate the hash operand from the has output

Hashrate -The number of cryptographic hashes performed per unit time

Openness – The openness of a target is the degree to which it can be accessed by entities other than the attacker

PBKDF - A password based key derivation function is a function that takes as an operand a normal user password operates on it to output a key to be used ,typically the key would be much harder to find than the password

Plaintext – The plaintext of a hash is the value that was passed to the hash function to produce it

Target – The target of a ciphertext is the plaintext that can be accessed by the decryption of the ciphertext. Normally the plaintext of the ciphertext itself, but in the case of web apps it may be password user accounts and functionality

Volatility – The volatility of a ciphertext is whether or not the guardian of the target has the means to remove access to the target

Symbols

A – The cryptographic hash algorithm used

α – The index of the storage contract in the root contract tree

ν_a – The Ethereum address of an account a

E – Electrical energy usage

i – The index of a cryptographic hash within CrackNet

L- The set of password lengths to try

N – Maximum number of hashes to compare

S – The index of the storage contract for a cryptographic hash within CrackNet

R – A CrackCoin reward

$R_{A,i,[\alpha]}$ – A CrackCoin reward at the challenge index I for algorithm A in a storage contract, optionally of index α in the root tree

C – A cryptographic hash

$C_{A,i,[\alpha]}$ – A cryptographic hash of ciphertext at the challenge index I for algorithm A in a storage contract, optionally of index α in the root tree

H – A cryptographic hash function, H(P)=C

$\omega_{A,\tau}$ – The number of hashes the cracker can perform per unit time for that algorithm A and attack mode τ.

P – Plaintext of a cryptographic hash

τ – A password cracking attack mode

Λ – Power used by the crackers machine

$\langle \Lambda \rangle$ -Average power used by the crackers machine

Γ - A correction cost to account for the potential differences between the theoretical and real world electricity costs

π – Cost per unit electrical energy used

Ψ - An extra reward on top of the base electricity cost set by the user

ξ–Minimum reward to cover the electrical energy costs

T-Total cost of a single cracking run

Ω– The time taken for a cracking machine to search a set of passwords

$\mu_{A,\tau,L}$ -The success rates for cracking challenges of specific algorthm A, with attacl mode τ, and length L

Introduction

In order to authenticate users most software applications and webservices utilise a password system, where the user must enter a specific string in order to gain access to the program or website. In addition passwords may be used to encrypt data directly or using password based key derivation functions (PBKDF).

A password is a defined string of bytes that when passed to a software entry guard as an operand enables access or further access to a program or service by a user.
User defined passwords are typically ASCII strings but these may be used to seed PBKDF functions for greater security.

In the case of authentication the software normally stores a cryptographic hash of the password so that the actual password cannot be read from the computer memory by an attacker.

Most cryptographic hash functions are virtually impossible to crack using a single algorithm with todays computing capability , if a user forgets their password unlike a web service thay cannot reset it, they must find the right password.
One method is brute forcing where an attacker literally attempts every feasible password until the right one is found. As expected this is a time consuming method ,the number of possible passwords increases exponentially with increasing password length for a given character set.

An sample password may be a lowercase alpha numeric password (a-z 0-9) with 37 possible characters and may be hashed with the popular algorithm SHA-256

Alpha numeric password a-z 0-9 A-Z

Number of characters	Number of possibilities	Estimated time to crack with a GTX 1080 Ti (avg hashrate 4636.7 MH/s)[6]
5	916e8	0.198 seconds
8	2.18e14	13 hours
15	7.69e26	5.26 billion years
20	7.04e35	4.82 quintillion years
50	4.16e89	2.85 trevigintillion years

Brute force password recovery is a normally resource intensive procedure, as such the average internet user attempting this for password recovery or security research may not be able to successfully perform this with their hardware.

Dedicated cracking systems typically use high end computers, many graphics card in parallel or supercomputers, these are expensive and require additional expertise to set up.

To account for this there are services online such as online hash/rainbow tables which can be searched for hashes and paid services run on single servers where users may pay crackers for plaintext.

Blockchains are among the largest reserves of active computing power on the public internet, the average daily hashrate on the Ethereum network is above 100,000 gigahashes per second for sha256[6].
The estimated amount of energy to be expended by the Ethereum miners in 2019 is estimated to be 7.1 terawatt hours [7] comparable to the country Bolivia and 73 terawatt hours for Bitcoin miners comparable to the country of Austria[8], for this the blockchain miners win some cryptocurrency reward.

Hence we can see large amounts of people and companies are already expending astronomical amounts money in the form of electrical energy costs and advanced hardware on blockchain applications.

Naturally one would wish to design an application that could harness this power for cracking password hashes, additionally the blockchain infrastructure provides the benefit of facilitating payments via trustless smart contracts.
This provides a financial incentive for users to spend money on electrical energy to crack hashes.

Using Ethereum based smart contracts CrackNet securely administrates hash storage and escrow of the cryptocurrency reward. CrackNet smart contracts then performs the comparison of any plaintext submitted and pays the reward to the cracker.

CrackNet is the simplest and easiest to use method for paying for hash plaintext, a user simply generates an Ethereum account, purchases CrackCoin and submits a hash with a reward.

Crackers online who own purpose built machines can then freely browse the CrackNet storage contracts, download hash challenges for the Ethereum node and attempt to crack the password.

Overview

CrackNet is a system of smart contracts currently deployed on the Ethereum blockchain using the cryptocurrency token CrackCoin(CRK), for the purpose of administrating the paying of crackers by internet users for cracking cryptographic hashes.

Using the CrackNet main contract family or single smart contracts, web3 users can store a cryptographic hash in the storage contracts and securely escrow a cryptocurrency reward.

To redeem the reward another user simply sends the plaintext of the hash to CrackNet redeem contracts which sends the reward to the cracker if the plaintext is correct.

This gives any internet user access to the power of any computer on the planet, without owning such assets themselves, thereby substantially increasing their agency.

CrackNet currently hashes the builtin Ethereum functions, SHA-256, KECCAK-256,RIPEMD160, but is capable of hashing practically any function. These will be added and the documentation will be updated.

In addition CrackNet has contracts for storing user profiles for statistics and for users to upload information and multimedia about their machine on the blockchain.

CrackNet does not place any additional fees for normal usage on top of the Ethereum gas costs.

CrackNet is built as a store of cryptographic hash challenges, via our client hash descriptors can be sent to a connected DCS.

Mission

The aim of this research project is to understand the capability of public unknown computing assets ,to identify the challenges in harnessing them and to innovate to overcome them.

In addition CrackNet is being developed to understand the capability of distributed public assets.

CrackNet itself may serve as a stepping stone to finding ways for single internet users to harness public computing power for other purposes.

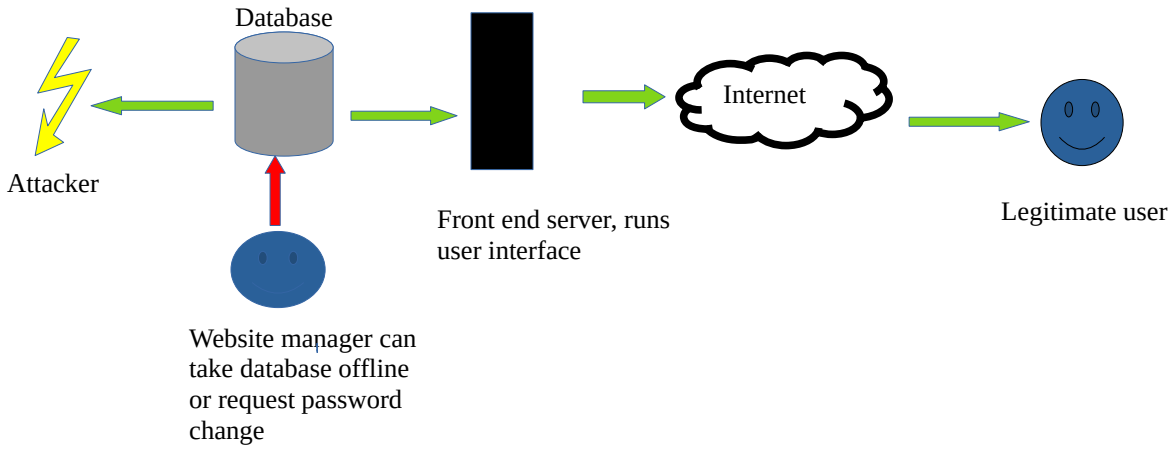
A distributed cracking system on Ethereum is being developed with smart contracts to increase the efficiency of the application and substantially reduce cracking time.

CrackNet vs Digital Secrets

Volatility

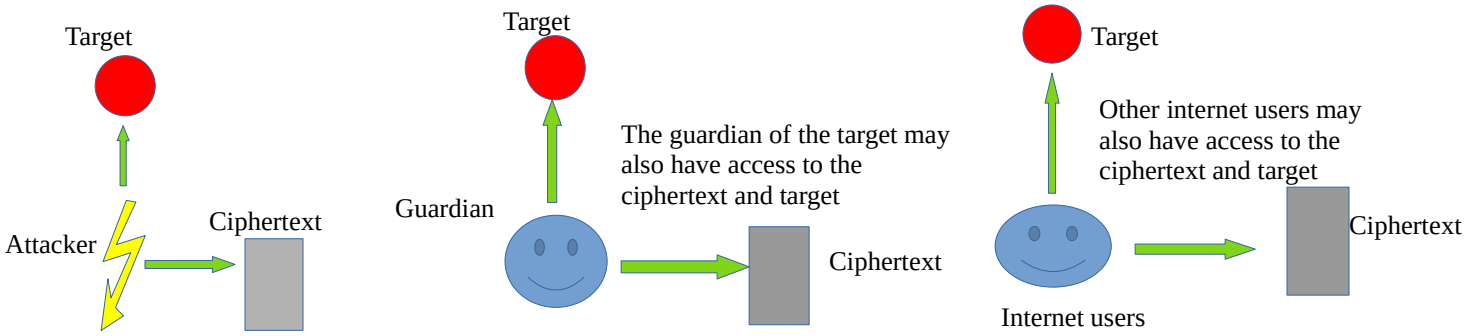
I define an volatile secret as cryptographic data where there is means for access to be restricted to the target should those responsible for it decide to do so. An example would be an authentication database on a company server. The volatile secrets are the users password hashes, the target could be financial information, personal information,access to users control panels , which the attacker would be able to access using the password.
If the web company were to discover that individuals were attempting to crack their hashes they could restrict access to the target by taking their server offline, remedying the security flaw or making users change their passwords.

Volitale hashes in an online password database



In the case of an involatile secret there is no means for those responsible to restrict access to the target, as such it may not necessarily matter to the attacker whether the secrets owner is aware of their activity .

Involatile cipher-text



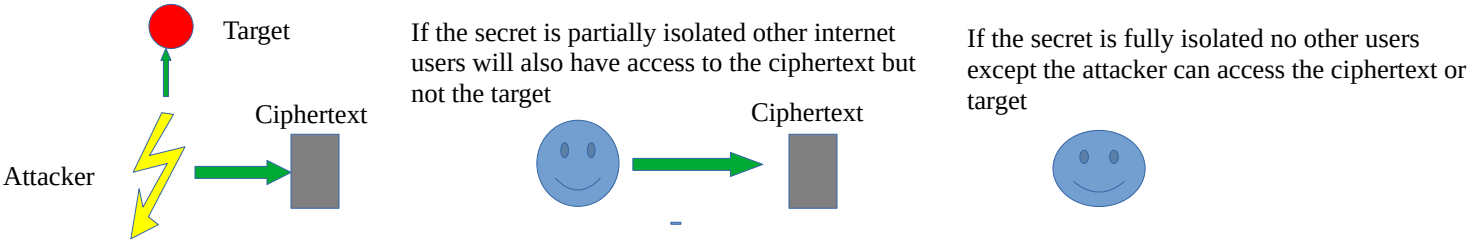
For example take an attack on a operating system user authentication hash , the attacker is performing password recovery to regain access to their machine. This is an involatile secret

and there is no way for access to the target (the users account on the operating system) to be restricted if the attackers activities become public.
There would be virtually no way for the secret owner to stop any member of the public by assisting the attacker by finding the plaintext.

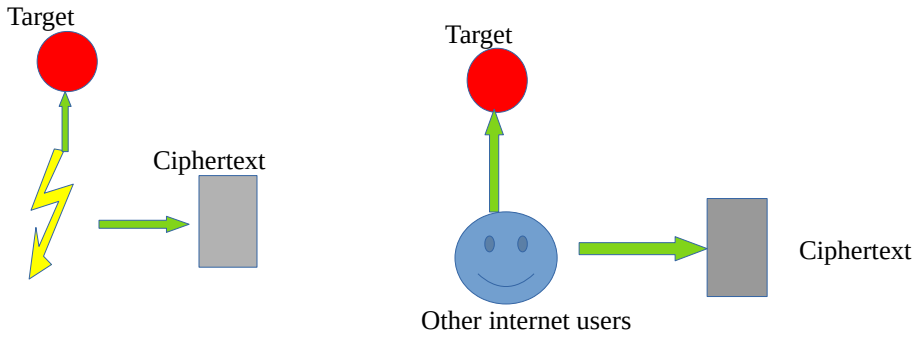
In the case of cracking a cryptographic hash to gain physical access to a machine this is a volatile secret if it is on premises belonging to the target or under their control, and involatile if the machine is in the custody of the attacker.

Isolation

An secret is partially isolated if there is no means for anyone except the attacker to gain access to the target data should they discover the plaintext.



I define an open secret as one where individuals other than the attacker may gain access to the target data , should they discover the plaintext.



Generally in CrackNet secrets will be partially isolated or open.

Criticality

A secret is critical if the attacker cannot have anyone else access the target data, and non-critical if it does not matter to the attacker if the public has access to the target data.

When open source CrackNet code is available users will be able to deploy CrackNet contracts onto a private blockchain for use against critical information.

Applicability of CrackNet

Volatile Secrets

	Isolated	Partially isolated	Open
Critical	-	Medium risk	High risk
Non critical	-	Medium risk	Medium risk

Involatile Secrets

	Isolated	Partially isolated	Open
Critical	Little risk	Medium risk	High risk
Non critical	No risk	No risk	No risk

Obviously CrackNet is most effective for use against isolated involatile secrets,

For example individual is attempting to crack a hash to recover access to an operating system on their computer, this is an isolated critical involatile secret, CrackNet is safe to use as long as the new password is immediately applied.

Given the sheer amount of webservices on the internet ,it is likely that attackers will find success using CrackNet against open volatile secrets, however without breaking the Ethereum blockchain, there would be no means for them to erase evidence of the act.

CRACKCOIN - The CrackNet Cryptocurrency

CrackNet is currently running on the Ethereum blockchain and uses CrackCoin (CRK) an ERC20 smart token.

CrackCoin - pre Alpha token sale

Mainnet Token Address	0x8Fe6736c0bb4bCbd7D8156963cB247F8a6b8d5a0
ICO smart contract	0x8573EFB7B522aeAaAEaD7791486C85eF5283ADD8
Sale amount	1,000,000 CRK
Decimals	18
ICO price	125 CRK=1 ETH
ICO currency Options	Ether

The ICO smart contract securely sends you CRK at the ICO price when you send it ether. When there are no tokens left the smart contract reverts the transaction and you get your ether back.

Rinkeby testnet token sale – pre Alpha token sale

Testnet token address	0x57BAC05773E552ba0DC1d0a0db27c74C40d5592D
ICO smart contract	0x8ed0C7257cC3644C13305aE07E5facc550894ba3
Sale amount	1,000,000 CRK
Decimals	18
ICO price	125 CRK = 1 ETH
ICO currency options	Ether

or you could get free testnet tokens in the telegram group: https://t.me/cracknetofficial by including your ethereum address in a genuine message.

Usage

CrackNet Basic Algorithm 1 (CNBA-1)

Uploading Hashes

To submit a hash to CrackNet the user sends a transaction with the function call newChallenge() on the storage contract, with the arguments as the algorithm, hash

1. User has a new cryptographic hash e.g. H='0x49596c34a195a4c3d893fa244f796a3d1473c0be36f20a760a6c4ead5329f382', A='sha256'
2. user sends the transaction with the function call newChallenge(A,H,R), R is the reward in CRK
3. The smart contract checks the input data and transfers the reward from the user account into escrow

Boosting Challenges

Individuals other than the poster may increase the cryptocurrency reward to increase the chance of it being cracked.

1. A user may increase the reward for a hash by ΔR
2. A user sends the transaction with the function call boostChal(A,i,ΔR)
3. The smart contract transfers the reward from the user account

Updating Values

Additionally the poster can update:

- the hash using the function setHash(A,i,C)
- the suspected salt using the function setSalt(A,i,S)
- the suspected plaintext length using the function setLength(A,i,L[0])

Submit Plaintext

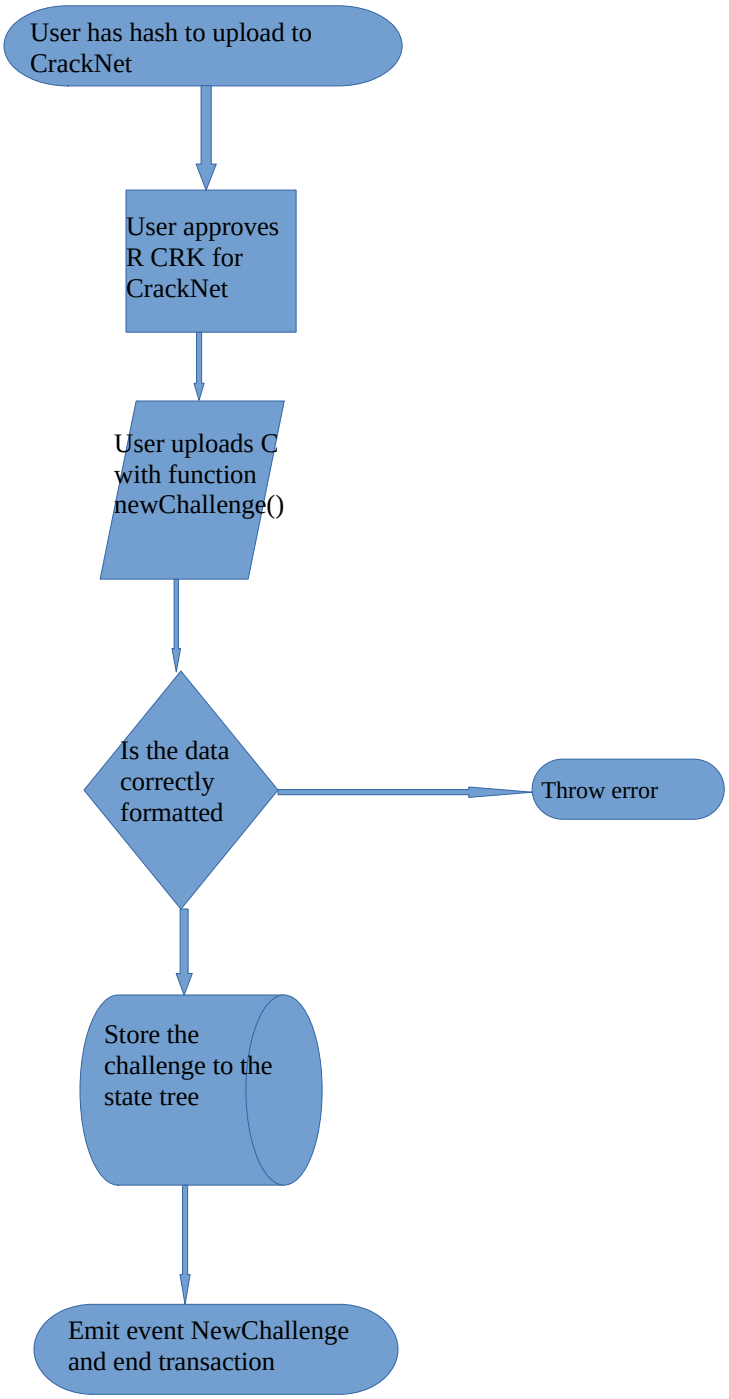
To claim the cryptocurrency reward the user must upload the plaintext to the smart contract system.

1. User with the Ethereum address E_u successfully cracks a C as plaintext P
2. E_u sends the transaction with the function submitCrack(A,i,α,P)
3. If $H(P)=H_{A,I}$ the smart contract pays $R_{A,I}$ to the address E_u

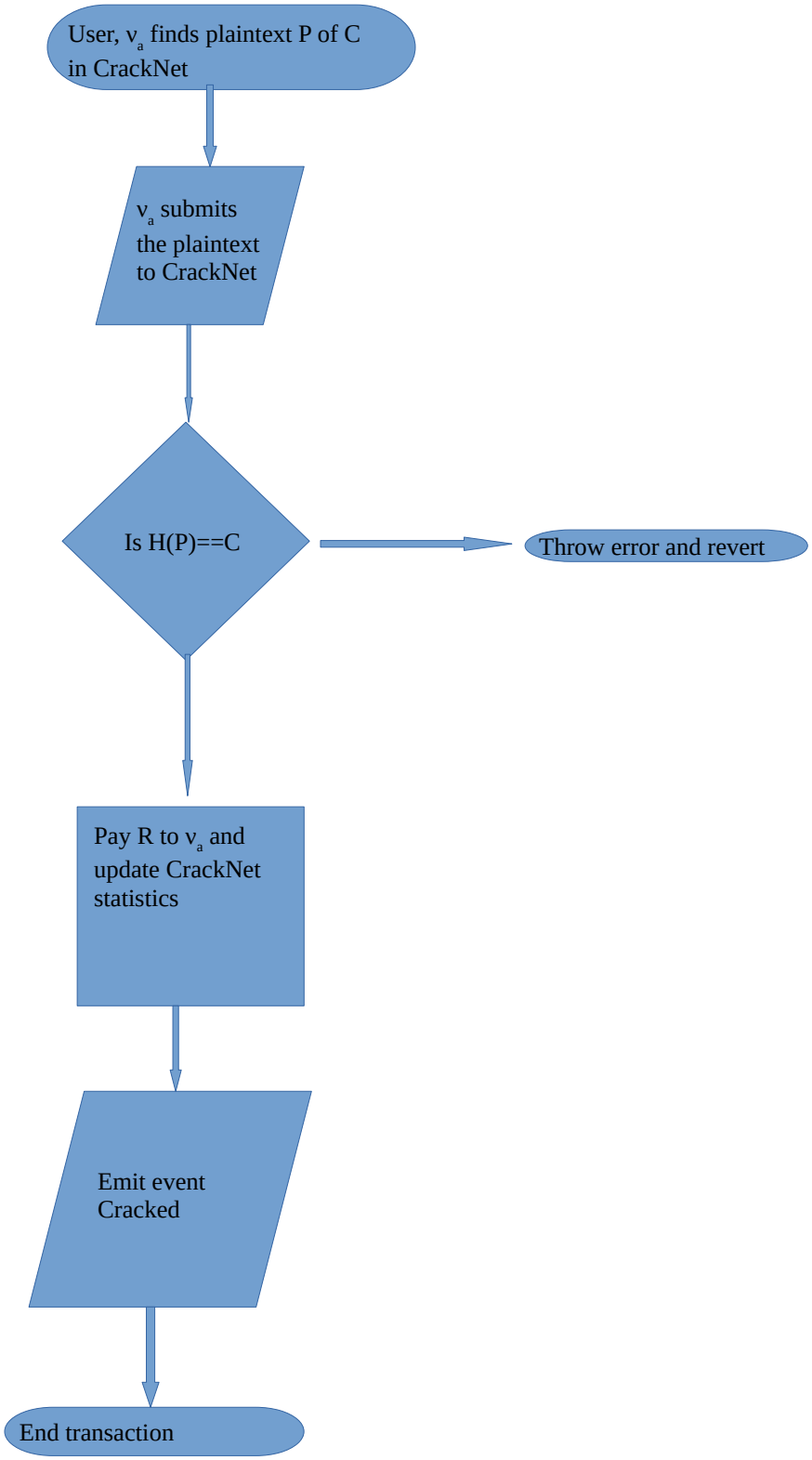
CrackNet Flowcharts

Upload a hash

A user has a hash H, which they upload for reward R.



Submit plaintext, using internal functions (sha256, keccak256, ripemd160)



Costs and profits

Now we will lay the foundations for a mathematical system to describe and help to predict costs and profits for the cracker using CrackNet.

Let us use the expected electrical power consumption to calculate the theoretical minimum price for a hash crack.

We will use an ether price of 1 ETH=200.00 USD which is normal at the time of writing.

The expectation of the time taken for a computing asset of hashrate $\omega_{A,\tau}$ (for the hashing algorithm A and attack method τ) to perform N comparisons is $\langle \Omega \rangle$

$$\langle \Omega \rangle = \frac{N}{\omega_{A,\tau}}$$

The maximum number of guess for passwords of lengths L bits may also be calculated as

$$N_{max} = \sum_{k=0}^{m} 2^{L_k}$$

But N_{max} may also be the total length of a wordlist or the maximum number of comparisons performed with another attack method. Also use $\hat{N}$ in the place of N_{max} .

Cost differences

A cracker runs a machine expends power $\Lambda(t)$, which can be experimentally obtained. For simplicity the cracker uses an expectation or average value of power used to derive the price $\langle \Lambda \rangle$ .This may be further specified with the parameters A and τ representing the cryptographic hash algorithm and attack mode respectively $\langle \Lambda_{A,\tau} \rangle$

The total electrical energy expended by a machine is

$$E = \int_0^{\Omega} \Lambda \, dt$$

where $\Lambda = \Lambda(t)$

When modelling to calculate energy costs we use the case where the cracker searches the entire set of passwords which would take the time Ω.
Using the expectation of power or average power the expectation of energy usage is

$$\langle E\rangle = \int_0^{\langle\Omega\rangle} \langle\Lambda_{A,\tau}\rangle dt = \langle\Lambda_{A,\tau}\rangle\langle\Omega\rangle$$

Therefore the expected cost of this energy is $T=\pi\langle E\rangle$, with π as the cost per unit electrical energy.
The difference in energy used between the real and average cases is

$$\Delta E = \int_0^{\Omega} \Lambda\, dt - \langle\Lambda_{A,\tau}\rangle\langle\Omega\rangle$$

The cost of the energy difference, $\Delta T=\pi\Delta E$.
Clearly when ΔT >0 the cracker may make a loss, and when ΔT<0 the cracker may a profit by overestimating the power usage.

Mean energy

The cracker may take samples of the energy expended by the machine during cracking.

With the energy used $E_k = \int_0^{\Omega} \Lambda\, dt$ and $\langle E_{A,\tau,N}\rangle = \frac{\sum_{k=0}^{M} E_k}{M}$.

The cost of electrical energy is $\langle T\rangle = \pi\langle E_{A,\tau,N}\rangle$, the cracker may therefore set a minimum price
$\xi = (1+\Gamma)\pi\langle E_{A,\tau,N}\rangle$.

Again we will get $\Delta E = \int_0^{\Omega} \Lambda\, dt - (1+\Gamma)\langle E_{A,\tau,N}\rangle$, again to satisfy $\Delta E \leqslant 0$ we must set

$$\Gamma > \frac{E}{\langle E_{A,\tau,N}\rangle} - 1 \quad .$$

If $E \sim \Phi(\langle E\rangle, ...)$, where Φ represents an arbitrary probability distribution with a width W
then $\langle E_{A,\tau,N}\rangle + \frac{W}{2} = (1+\Gamma)\langle E_{A,\tau,N}\rangle$ will ensure no losses. Or we can set

$$\Gamma = \frac{W}{2\langle E_{A,\tau,N}\rangle} \quad .$$

Mean power

The cracker may base the price on the mean power used by the machine, in a sample of M samples.

$$\langle \Lambda_{A,\tau} \rangle = \frac{\sum_{k=0}^{M} \Lambda_k}{M}$$

The cracker may set a minimum price for users to be
$\xi = (1+\Gamma)\langle \Lambda_{A,\tau} \rangle \langle \Omega \rangle \pi$
to account for differences between the real and theoretical.

The cost of this difference is

$$\Delta T = \pi\left(\int_0^{\Omega} \Lambda \, dt - \langle \Lambda_{A,\tau} \rangle \langle \Omega \rangle (1+\Gamma)\right)$$

The minimum Γ that can be used is where ΔT=0 or

$$\Gamma = \frac{1}{\langle \Lambda_{A,\tau} \rangle \langle \Omega \rangle} \int_0^{\Omega} \Lambda \, dt - 1$$

or

$$\Gamma = \frac{\omega_{A,\tau}}{\langle \Lambda_{A,\tau} \rangle N_{max}} \int_0^{\Omega} \Lambda \, dt - 1$$

or for all possible password guesses

$$\Gamma = \frac{\omega_{A,\tau}}{\langle \Lambda_{A,\tau} \rangle \sum_{k=0}^{m} 2^{L_k}} \int_0^{\Omega} \Lambda \, dt - 1$$

again if $\Lambda_{A,\tau}$ ~ $\Phi(\langle \Lambda_{A,\tau} \rangle, ...)$ with a width W .

Then we can set $\Gamma = \frac{W}{2\langle \Lambda_{A,\tau} \rangle}$.

Estimating time to complete

In prediction calculations we use $\langle\Omega\rangle=\frac{N_{max}}{\omega_{A,\tau}}$, if the password is not found then $\Omega\approx\langle\Omega\rangle(1+\Gamma)$, in this power calculation.

In a single run a cracker may perform N_{run} comparisons so over a time λ , as our simple model uses $\langle\Omega\rangle$ derived from N_{max}, we must calculate the expectation of the time taken to complete the search had the password not been found

$$\Omega_{predicted}=\frac{N_{max}}{N_{run}}\times\lambda$$

which may be used to correct the minimum price as

$$\Gamma=\frac{1}{\langle\Lambda_{A,\tau}\rangle\Omega_{predicted}}\int_0^{\Omega_{predicted}}\Lambda\,dt-1$$

In the case where $N_{run}=N_{max}$, then there is no error and $\Omega=\Omega_{predicted}$.

A user will normally add an additional reward Ψ on top of the minimal price for cracking ξ to make it more likely for them to have their hash cracked.
In all the total reward for a hash challenge set by the user will be $R=\xi+\Psi$.
With Ψ as an additional reward for the cracker, any internet user may increase Ψ to $\Psi+\Delta\Psi$ by using the function boostChal(A,I,ΔΨ) on the CrackNet smart contract.

The cracker may also set Ψ_{min} for profit and to cover hardware and other costs.
The cracker will therefore make a net profit $\epsilon=R-T=\Psi-\Delta T$.

Real and predicted earnings

Of course a cracker will not manage to solve every cryptographic hash they try, therefore we must calculate an expected profit based on the success rates and electricity costs of the cracker.
A cracker completes searching M challenges $C_1+C_2+C_3+...C_M$ in a sample time $\hat{\Omega}=\sum_{k=0}^{M}\Omega_k$.

The total electrical energy cost is $T=\pi\sum_{k=0}^{M}E_k$ and the maximum available reward $\sum_{k=0}^{M}R_k$

The crackers earnings before energy cost deductions is $\chi=R$ if the cracker finds the plaintext and his earnings per unit time is $Z=\frac{\chi}{\hat{\Omega}}$

The cracker will make an earning $\chi=\sum_{k=0}^{M}\chi_k$ and profit $\epsilon=\sum_{k=0}^{M}\chi_k-T$.

Now let us look at the case in the predictive sense.

The cracker will have experimental rates of success with challenges specified as $\mu_{A,\tau,L}$ where $\mu_{A,\tau,L}=\frac{Number\ of\ successful\ cracks}{Number\ of\ challenges}$.

Or $\mu=\mu_{A,\tau}(N)$ using an appropriate function fitting.

Therefore for a set of M challenges $C_{A,\tau,L,k}$, we have $\langle\chi_k\rangle=\mu_{A,\tau,L}R_k$, here we have assumed the dependence of μ on R is negligible.

There is a residual $\delta=\chi_k-\langle\chi_k\rangle$, the superior theoretical model minimizes δ .

Or for an arbitrary set of M challenges C_i with μ_k, R_k we get $\langle\chi\rangle=\sum_{k=0}^{M}\mu_k R_k$ and $\langle Z\rangle=\frac{\langle\chi\rangle-\langle T\rangle}{\hat{\Omega}}$.

Our expected costs for energy $\langle T\rangle$ are calculated from the expected energy usage $\langle E\rangle$.

$\langle E\rangle=\sum_{k=0}^{M}\langle\Omega\rangle_k\langle\Lambda\rangle_k\Rightarrow\langle T\rangle=\pi\sum_{k=0}^{M}\langle\Omega\rangle_k\langle\Lambda\rangle_k$, or $\langle E\rangle=\sum_{k=0}^{M}\frac{\hat{N}_k}{\omega_k}\langle\Lambda_k\rangle\Rightarrow\langle T\rangle=\pi\sum_{k=0}^{M}\frac{\hat{N}_k}{\omega_k}\langle\Lambda_k\rangle$

One could also calculate $\langle Z\rangle=\frac{\langle\chi\rangle-\langle T\rangle}{\langle\hat{\Omega}\rangle}$, using $\langle\hat{\Omega}\rangle=\sum_{k=0}^{M}\langle\Omega\rangle_k$.

Further analysis could lead to choosing appropriate probability distributions for the number of comparisons to crack real passwords for specific algorithms and attack modes.

As such we may use $\Omega_{A,\tau}\sim\Phi_{A,\tau}(L)$ as the probability distribution.

To calculate the profit $\epsilon=\chi-T=\Psi-\Delta T$ and on average $\langle\epsilon\rangle=\langle\chi\rangle-T$.

The cracker may therefore choose the challenges to try where $\langle\chi\rangle>\langle T\rangle$ or $\langle\chi\rangle>\pi\sum_{k=0}^{M}\langle\Lambda\rangle_k\langle\Omega\rangle_k$ to satisfy $\langle\epsilon\rangle>0$.

Example

Using data from Patrick Kennedys 8x GTX 1080 Ti test [], lets model an attack on an 8 character alphanumeric (0-9a-zA-Z) password.

$\langle \Lambda \rangle$	2659W
$\omega_{SHA\,256,\,bruteforce}$	37093.2 MH/s
π	13.41 ¢/kWh []
N_{max}	2.18×10^{14}
A	SHA256

For convenience recalculate $\pi = 3.725\times10^{-6}$ ¢/J

The total cost $T = \frac{2.18\times10^{14}\times2659\times3.725\times10^{-6}}{3.70932\times10^{10}} = \$0.58$, here we have made the unlikely assumption that power usage is constant, we used the mean power to calculate energy costs.

(based on Californian commercial rates where the cracking test was performed)

Using an initial Γ of 10% we have $\xi = \frac{1.1\times2.18\times10^{14}\times2659\times3.725\times10^{-6}}{3.70932\times10^{10}} = \$0.64$

So the cracker sets ξ= $0.64=0.4 CRK as the price for energy costs.

Maximum time for the cracker to search the password set

$$\langle \Omega \rangle = \frac{N_{max}}{\omega_{SHA\,256}} = 98\,minutes$$

This is ignoring the gas costs for the user for adding to the EVM state tree by submitting the challenge to CrackNet.

In development this has been around 100000 gas in Ganache currently at a 20 Gwei gas price so 2×10^{15} ether or $0.40.
In total $1.04 minimum spending on cracking this password for the user.
Therefore we can say that this type of transaction is definetely feasible and profitable and that clearly this hardware may be better suited for cracking for profit rather than mining for profit.

The average energy cost (not net income) per day is therefore from this instance :

$$\langle T \rangle = \frac{0.58}{98} \times 1440 = \$8.52$$

I expect a minimum total reward to be $5, hence Ψ=$3.96, the rig would have to crack 3 such challenges per day to make profit which would take 294 or around a fifth of the day.

By comparison look at the case where we use the 8 GTX 1080 Ti rig for mining Ethereum.

An estimation of the mining hashrate for the 8 GTX 1080 Ti rig is 272 MH/s [10], using the electricity cost $\pi = \$0.1341/kWh$ we calculate losses of up to $2.29 per day [11] for a total power consumption of 2000W and ether price of $200.00.

Therefore we can say that given enough feasible passwords CrackNet could be much more profitable than mining Ethereum using minimal values.

Variation in cryptocurrency price

The cryptocurrency price may increase or decrease during the time in which a hash challenge is unsolved.

Both the users and the crackers want the price of CrackCoin to increase. If the price falls too low the users challenge will become less profitable and so the chance of being cracked will decrease unless the user boosts the challenge by adding more cryptocurrency.
The cracker will make more profit of plaintexts if the CrackCoin price increases, they could even hold plaintexts to claim later when the price is higher with the risk of another entity finding the plaintext.

Future

CrackNet is being developed to be used with the any cracking program the cracker wishes to use, therefore CrackNet is able to work with most hashing functions using the system we are developing, and will be expanded to immediately work with MD5,MD4 ,SHA-512,SHA-1 etc ..

The Beta version of CrackNet will use an advanced optimization for faster password recovery. In the near future I aim to release some more open source code to enable its deployment on private blockchains.

I also intend to collect benchmark data hashrates and power consumption on different GPUs and CPUs using funds received.

Appendix A - Algorithms currently supported by CrackNet

Algorithm	Algorithm Identifier	Facilitator
SHA-256	“sha256”	EVM inbuilt function
Keccak-256	“keccak256”	EVM inbuilt function
ripemd-160	“ripemd1660”	EVM inbuilt function

The beta release will add many more algorithms.

References

1. Narayanan, Arvind; Bonneau, Joseph; Felten, Edward; Miller, Andrew; Goldfeder, Steven (2016). Bitcoin and cryptocurrency technologies: a comprehensive introduction. Princeton: Princeton University Press. ISBN- 978-0-691-17169-2
2. Andy Greenberg (20 April 2011). "Crypto Currency". Forbes. Archived from the original on 31 August 2014. Retrieved 8 August 2014
3. Cryptocurrencies: A Brief Thematic Review Archived 25 December 2017 at the Wayback Machine. Economics of Networks Journal. Social Science Research Network (SSRN). Date. Retrieved 28 August 2017.
4. Schueffel, Patrick (2017). The Concise Fintech Compendium. Fribourg: School of Management Fribourg/Switzerland. Archived from the original on 24 October 2017. Electricity Local,California Electricity Rates and Consumption, 2012 ,https://www.electricitylocal.com/states/california//
5. Wikipedia, Hash function, 2019, https://en.wikipedia.org/wiki/Hash_function
6. Patrick Kennedy, Password Cracking with 8x NVIDIA GTX 1080 Ti GPUs ,2017 ,https://www.servethehome.com/password-cracking-with-8x-nvidia-gtx-1080-ti-gpus/
7. Etherscan.io,Average daily hashrate,2019,https://etherscan.io
8. Alex de Vries,Ethereum Energy Consumption Index,2019,https://digiconomist.com
9. Alex de Vries, Bitcoins Growing Energy Problem,2018, Joule, Cellpress
10. MiningChamp ,Hashrate of Graphics Card ,2019 ,https://miningchamp.com/
11. Coinwarz.com ,Ethereum Mining Calculator and Profit Calculator ,2019 ,https://www.coinwarz.com/calculators/ethereum-mining-calculator/?h=272&p=2000&pc=0.1341&pf=0.00&d=2269032633443130&r=2.00000000&er=200&btcer=1&hc=0.00