Breaking and Fixing Gridcoin

Martin Grothe, Tobias Niemann, Juraj Somorovsky, Jörg Schwenk
Horst Görtz Institute for IT Security, Ruhr University Bochum
{firstname.lastname}@rub.de

Abstract

Bitcoin has been hailed as a new payment mechanism, and is currently accepted by millions of users. One of the major drawbacks of Bitcoin is the resource intensive Proof-of-Work computation. Proof-of-Work is used to establish the blockchain, but this proof of work does not bring any benefits and arguably is a waste of energy. To use these available resources in a more meaningful way, several alternative cryptocurrencies have been presented. One of them is Gridcoin, which rewards the users for solving BOINC problems. Gridcoin currently possesses a market capitalization of $ 23,423,115.

In our work we conducted the first security analysis of Gridcoin. We identified two critical security issues. The first issue allows an attacker to reveal all email addresses of the registered Gridcoin users. Even worse, the second issue gives an attacker the ability to steal the work performed by a BOINC user, and thus effectively steal his Gridcoins. These attacks have severe consequences and completely break the Gridcoin cryptocurrency.

We practically evaluated and confirmed both attacks, and responsibly disclosed them to the Gridcoin maintainers. We developed backwards compatible design changes for the Gridcoin system, in order to protect users' trust into this promising approach.

Keywords: Gridcoin, Attack, BOINC, Proof-of-Stake, Meaningful Cryptocurrency

1 Introduction

Since the launch of Bitcoin in 2009 [18, 10], the new peer-to-peer electronic cash system grew to be one of the few widely accepted payment methods in both online and retail shopping. The release of the official Bitcoin Core client, developed by the Bitcoin inventor Satoshi Nakamoto as open source software, allowed users to fork the project and apply their own design changes. Hence we could observe emergence of many new Bitcoin based cryptocurrencies, so called altcoins. These altcoins attempt to establish new user communities by advertising different technology improvements or tweaks. Nowadays, about 700 Bitcoin based cryptocurrencies exist [4].

Wasteful computation. One of the main drawbacks of Bitcoin (and other Bitcoin based cryptocurrencies) is the resource expensive Proof-of-Work computation. Proof-of-Work is used to establish the blockchain, which contains blocks with Bitcoin transactions. In order to construct a new valid block, the so called miners collect Bitcoin transactions. They attempt to hash data from the previous block together the transactions and some randomness so that the resulting SHA-256 value lies below a specific boundary (i.e., the hash value starts with a specific number of zeros). The search for a valid block results in many SHA-256 computations and thus requires a huge amount of energy. High rewards for the block computations, however, motivate many users to participate in the mining process. Increasing the overall security on the one hand, on the other a study from 2014 suggests that *Bitcoin's total electrical footprint is comparable with the Irish national electrical energy consumption* [19].

Cryptocurrencies with meaningful computations. The energy spent for Bitcoin mining is arguably wasteful since it results in meaningless hash computations. This fact has motivated many developers to establish new altcoins. Primecoin introduces a new Proof-of-Work scheme based on computation of prime number chains – Cunningham chains [1]. Filecoin and Permacoin introduce a concept based on proof-of-retrievability [5, 6], which rewards users for providing data storage. This way it is possible to create a distributed file storage systems. Wustrow and VanderSloot showed that it is possible to create a cryptocurrency based on a malicious Proof-of-Work [31]. Their DDoSCoin concept rewards miners for contributing to a distributed Denial-of-Service (DoS) attack. Nevertheless, these altcoins only attempt to solve

specific problems and their concepts are not applicable in general.

Gridcoin. Gridcoin is an altcoin, which is in active development since 2013. It claims to provide a high sustainability, as it has very low energy requirements in comparison to Bitcoin. It rewards users for contributing computation power to scientific projects, published on the BOINC project platform. Although Gridcoin is not as widespread as Bitcoin, its draft is very appealing as it attempts to eliminate Bitcoin's core problems. It possesses a market capitalization of $ 23,423,115 [3] (11th of May 2017) and its users contributed approximately 5% of the total scientific BOINC work done before October, 2016 [2]. This motivates for the analysis of this relatively new cryptocurrency.

Security analysis of Gridcoin. In this paper we analyze the security of Gridcoin. We first give an overview of the Gridcoin core security functionalities and its connection to the BOINC network. In particular, Gridcoin uses the BOINC statistics servers to reward BOINC users for their computational contributions to solving meaningful problems. The first general issue arises when considering the trust into the statistics server. If the attacker is able to take control over this server, he is able to create an arbitrary number of Gridcoins. However, it is not necessary to have such a powerful attacker with server administration rights to execute practical attacks on Gridcoin. In this paper we demonstrate a critical vulnerability in Gridcoin, which puts the currencies market capitalization at risk. Our attacker is able to steal Gridcoins from legitimate Gridcoin users. We provide a proof of concept exploit implementation, which makes it possible to generate illegitimate Gridcoins. Our findings break the altcoin Gridcoin in the state of version 3.5.8.6 and thereby affect all Gridcoin users and exchanges.

Contributions. This work makes the following contributions:

- We give the first in depth description of the Gridcoin architecture and functionality.
- We analyze the security of Gridcoin and discuss the trust issues of this cryptocurrency.
- We present two feasible attacks on Gridcoin. First, our attacker can affect Gridcoin privacy and extract all email addresses of registered users. Second, our attacker can even steal computation power performed by Gridcoin users, and thus steal their Gridcoins.
- We developed backwards compatible design changes against the presented attacks.

Responsible disclosure. We responsibly disclosed our security findings to the Gridcoin maintainers on the 16th of September 2016 together with our proposed countermeasures. They responded to us that they are going to fix the vulnerabilities. Unfortunately, our proposals were not implemented correctly and further security issues were introduced with the new releases of Gridcoin. We contacted the maintainers again in order to help them fix the issues, but they did not respond to our following contact attempts. Thus, the current mandatory release 3.5.9.6 still contains several security vulnerabilities.[1]

2 BOINC

To solve general scientific meaningful problems, Gridcoin draws on the well known *Berkeley Open Infrastructure for Network Computing* (BOINC). It is a software platform for volunteer computing, initially released in 2002 and developed by the University of California, Berkeley. It is an open source software licensed under the GNU Lesser General Public License. The platform enables professionals in need for computation power to distribute their tasks to volunteers. Nowadays it is widely used by researchers with limited resources to solve scientific problems, for example, healing cancer, investigate global warming, finding extraterrestrial intelligence in radio signals and finding larger prime numbers.

When launching a BOINC project, its maintainer is required to setup his own BOINC server. Project volunteers may then create accounts (by submitting a username, a password and an email address) and work on specific project tasks, called workunits. The volunteers can process the project tasks and transfer their solutions with a BOINC client. The BOINC client is controlled via its RPC interface using either the GUI *boincmgr* or the CLI *boinccmd*. The client offers a broad variety of options to limit the shared resources, such as by timer or processing unit workload, and thereby gives the volunteer a lot of control and the ability to run it seamlessly in the background. The BOINC project server usually deploys executables for Windows, Mac OS X, Linux or Android systems on x86, x86_64 or ARMv7. To measure how much work a machine has done BOINC implements a computation credit system. This system enables user competition and team establishment. The statistics are retrievable from the BOINC project servers as XML data. To streamline the process, BOINC Credit statistics websites like boincstats.com[2] gather statistics from all BOINC project servers and process them for display in HTML markup, including graphs and rankings [27].

[1] For more details see: http://gridcoin-attacks.org

[2] `http://boincstats.com/`

2.1 BOINC Architecture

BOINC uses a client server architecture to achieve its rich feature set. The server component handles the client requests for workunits and the problem solutions uploaded by the clients. The solutions are validated and assimilated by the server component. All workunits are created by the server component and each workunit represents a chunk of a scientific problem which is encapsulated in an application. This application consists of one or multiple in-/output files, containing binary or ASCII encoded parameters. The BOINC server itself needs five daemons to satisfy the BOINC workflow [20].

1. **Scheduler** processes requests of the BOINC clients and estimates their computation power. It assigns each client an appropriate amount of workunits. All messages are XML formatted and transmitted via HTTP or HTTPS.
2. **Validator** validates the received results and assigns a respective amount of BOINC Credits, if the client sent valid results.
3. **Assimilator** transfers correct results into an external database or filesystem. If none of the clients is able to compute a validating result, the assimilator sends or writes error reports [20].

The description of the other two daemons is not relevant to our attacks, interested readers are referred to [20, 23].

2.2 BOINC Terminology

iCPID The BOINC project server creates the internal Cross Project Identifier (iCPID) as a 16 byte long random value during account creation. This value is stored by the client and server. From this point on it is part of every request and response between client and server [22].

eCPID The external Cross Project Identifier (eCPID) serves the purpose of identifying a volunteer across different BOINC projects without revealing the corresponding email address. It is computed by applying the cryptographic hash function MD5 to (iCPID, email) and thus has a length of 16 byte [22].

$$eCPID = MD5(iCPID \| email) \tag{1}$$

Furthermore, BOINC uses internal and external host identifiers, which are not relevant for the Gridcoin architecture and our attacks.

2.3 BOINC Credit

BOINC credits are generated whenever a host submits a solution to an assigned task. They are measured in Cobblestone, whereas one Cobblestone is equivalent to $\frac{1}{200}$ of CPU time on a reference machine with 1,000 mega floating point operation per seconds [21]. The served statistic files contain the two credit measuring values Total Credit and Recent Average Credit for each participant, host and team.

Total Credit: The value Total Credit is the total number of Cobblestones generated [21].

Recent Average Credit (RAC): RAC is defined as the average number of Cobblestones per day generated recently [21]. If an entire week passes, the value is divided by two. Thus old credits are weakly weighted. It is recalculated whenever a host generates credit [28].

$$RAC = RAC_{old} \cdot d(t) + (1 - d(t)) \cdot credit(new) \tag{2}$$

Where d(t) is the decay function with t being the time in seconds passed since the last RAC recalculation.

$$d(t) = e^{(-\ln(2) \cdot t / 604800)} \tag{3}$$

TeamRAC Is the average cumulative RAC of a BOINC team for a specific project.

3 Gridcoin-Research

Gridcoin was first introduced in 2013 as a fork of the alternative cryptocurrency Litecoin. Its primary goal was to reward reasonable scientific calculations performed via BOINC with coins. Therefore it replaced the Proof-of-Work concept of Litecoin with its own extended hybrid Proof-of-Work concept, called Proof-of-BOINC [13]. The original Gridcoin project was retired and forked due to massive changes in 2015. Besides the project was renamed to Gridcoin-Classic [24]. Our analysis focuses on Gridcoin-Research, the successor of Gridcoin-Classic. In this paper we refer to Gridcoin-Research as Gridcoin. The new fork introduced the Proof-of-Research concept to convert BOINC project contributions into extra currency shares.

3.1 Core Concepts

As a fork of Litecoin, Gridcoin-Research is a blockchain based cryptocurrency and shares many concepts with Bitcoin. While Bitcoin's transaction data structure and concept is used in an unmodified version, Gridcoin-Research utilizes a slightly modified block structure, which is shown abstractly in Figure 1. A Gridcoin-Research block encapsulates header and body. The header contains needed meta information and the body encloses transactions. Due to the *hashPrevBlockHeader*

field, which contains the hash of the previous *blockheader*, the blocks are linked and form the distributed ledger, the blockchain.

Blocks in the blockchain are created by so called *minters*. Each block stores a list of recent transactions in its body and further metadata in its header. To ensure that all transactions are confirmed in a decisive order, each *blockheader* field contains a reference to the previous one. To regulate the rate in which new blocks are appended to the blockchain and to reward BOINC contribution, Gridcoin-Research implements another concept called Proof-of-Research. Proof-of-Research is a combination of a new overhauled Proof-of-BOINC concept, which was originally designed for Gridcoin-Classic and the improved Proof-of-Stake concept, inspired by alternative cryptocurrencies [29, 15].

3.2 Proof-of-Stake

Proof-of-Stake is a Proof-of-Work replacement, which was first utilized by the cryptocurrency Peercoin in 2012 [15]. This alternative concept was developed to showcase a working Bitcoin related currency with low power consumption. Therefore the block generation process has been overhauled. To create a new valid block for the Gridcoin blockchain the following inequality has to be satisfied:

$$SHA256(SHA256(kernel)) < Target \cdot UTXO\ Value + RSAWeight \quad (4)$$

The kernel value represents the concatenation of the parameters listed in Table 2. The referenced unspent transaction output (UTXO) must be at least 16 hours old. The so called RSAWeight is an input value to the kernel computation, it's indicates the average BOINC work, done by a Gridcoin minter.

In direct comparison to Bitcoin's Proof-of-Work concept, it is notable that the hash of the previous *blockheader* is not part of the kernel. As a result it is theoretically possible to create a block at any point of time in the past. To prevent this, Gridcoin-Research creates fixed interval checkpoint blocks. Once a checkpoint block is synchronized with the network, blocks with older timestamps are invalid. Considering the nature of the used kernel fields, a client with only one UTXO is able to perform a hash calculation each time nTime is updated. This occurs every second, as nTime is a UNIX timestamp. To be able to change the txPrev fields and thereby increase his hash rate, he needs to gain more UTXO by purchasing coins. Note that high UTXO and RSAWeight values mitigate the difficulty of the cryptographic puzzle, which does increase the chance of finding a valid kernel. RSAWeight is explained in Section 3.3. Once a sufficient kernel has been found the referenced UTXO is spent in a transaction to the creator of the block and included in the generated block. This consumes the old UTXO and generates a new one with the age of zero.

The Gridcoin-Research concept does not require much electrical power, because the maximum hash rate of an entity is limited by its owned amount of UTXOs with suitable age. A 51% Attack, as it is possible in Bitcoin [26], is, nevertheless, still possible. For this an adversary would need to be in possession of 51% of all coins [25].

3.3 Proof-of-Research

Minters relying solely on the Proof-of-Stake rewards are called *Investors*. In addition to Proof-of-Stake, Gridcoin gives minters a possibility to increase their income with Proof-of-Research rewards. The Proof-of-Research concept implemented in Gridcoin-Research allows the minters to highly increase their block reward by utilizing their BOINC Credits. In this case the minter is called *Researcher*.

To reward BOINC contribution, relevant BOINC data needs to be stored in each minted block. Therefore, the software uses the *BOINCHash* data structure, which is encapsulated in the first transaction of each block. The structure encloses the fields listed in Table 6. The minting and verification process is shown in Figure 3 and works as follows:

1. A minter (*Researcher*) participates in a BOINC project A and performs computational work for it. In return the project server increases the users *Total Credit* value on the server. The server also stores the minter's *mail address*, *iCPID*, *eCPID* and *RAC*.
2. Statistics websites contact project server and download the statistics for all users from the project server (A).
3. After the user earned credits, his *RAC* will increase. Consequently, this eases finding a solution for the Proof-of-Stake cryptographic puzzle and the user will create (mint) a block and broadcast it to the Gridcoin network.
4. Another minter (*Investor* or *Researcher*) will receive the block and validate it. Therefore, he extracts the values from the *BOINCHash* data structure inside the block.
5. The minter uses the eCPID from the *BOINCHash* to request the RAC and other needed values from a statistics website and compares them to the data extracted from the *BOINCHash* structure, in case they are equal and the block solves the cryptographic puzzle, the block is accepted.

The following parameters, included in each *BOINCHash*, are relevant to our paper:

eCPID Identifier value of the researcher used in BOINC.

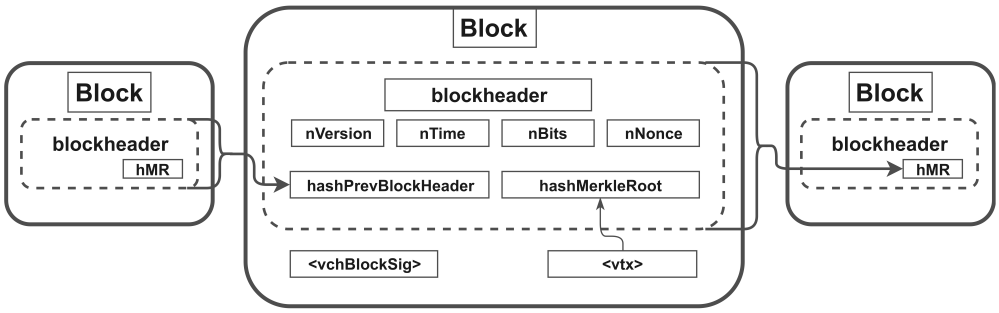

Figure 1: Gridcoin-Research block structure. Note that just a few header fields go into kernel computation

Field	Description
RSAWeight	Research Savings Account Weight
txPrev.block.nTime	block time of referenced tx
txPrev.nTime	timestamp of referenced tx
txPrev.vout.hash	hash of referenced tx output
txPrev.vout.n	index of refenced tx output
nTime	current UNIX timestamp

Figure 2: Gridcoin Kernel Parameter

GRCAddress contains the payment address of the minter.

ResearchMagnitudeUnit contains the Magnitude Unit. This parameter is calculated and used to keep a stable Proof-of-Research coin generation rate of 50,000 coins per 1000 blocks [25].

AverageMagnitude is the sum of project Magnitudes of a eCPID and represents the minters relative BOINC contribution. The project Magnitude is calculated with the following equation for every active BOINC project.

$$pMagnitude = \frac{RAC}{TeamRAC \cdot \#WhitelistedProjects} \quad (5)$$

ResearchAge is defined as the time span between the creation time of the last Proof-of-Research generated block with the user's eCPID and the time stamp of the last block in the chain measured in days.

$$ResearchAge = LastBlocknTime - LastPaymentTime \quad (6)$$

RSAWeight estimates the user's Gridcoin gain for the next two weeks, based on the BOINC contribution of the past two weeks.

ResearchSubsidy2 contains the calculated Proof-of-Research reward for the solved block.

InterestSubsidy encloses the computed Proof-of-Stake reward for the solved block.

CPIDv2 contains a checksum to prove that the minter is the owner of the used eCPID. We fully describe the content of this field in Section 6.

3.4 Reward Calculation

The total reward for a solved block is called **Subsidy** and computed as the sum of the Proof-of-Research and the Proof-of-Stake reward. The Proof-of-Stake reward is calculated with the following formula[3]:

$$PoSReward = \frac{Coinage \cdot 33 \cdot AnnualInterest}{365 \cdot 33 + 8} \quad (7)$$

In which **CoinAge** is the age of the referenced UTXO.

If a minter operates as an Investor (without BOINC contribution), the eCPID is set to the string *Investor* and all other fields of the *BOINCHash* are zeroed. An Investor receives only a relatively small Proof-of-Stake reward.

The profitable Proof-of-Research reward is computed as

$$PoRReward = ResearchAge \cdot AverageMagnitude \cdot ResearchMagnitudeUnit \quad (8)$$

As the Proof-of-Research reward is much higher than its Proof-of-Stake counterpart, contributing to BOINC projects is worth the effort.

4 Trust Issues and Threats

One critical aspect of cryptocurrencies is trust. For example, if a central exchange place (e.g., Mt. Gox) for a cryptocurrency is taken over, the exchange rate significantly decreases [17] and as a result the currency loses many of its users. This can also happen if a vulnerability in cryptocurrency is found. Ethereum's market capitalization decreased from $ 1.742 Billion US to $ 0.865 billion US, after two vulnerabilities were exploited by attackers on the 16th of June 2016.

In this section we show that besides the security flaws there are two major trust issues in Gridcoin the users should be aware of.

[3]Explanation of the numbers: http://wiki.gridcoin.us/Proof-of-Research#Proof-of-Research_Reward_Calculation

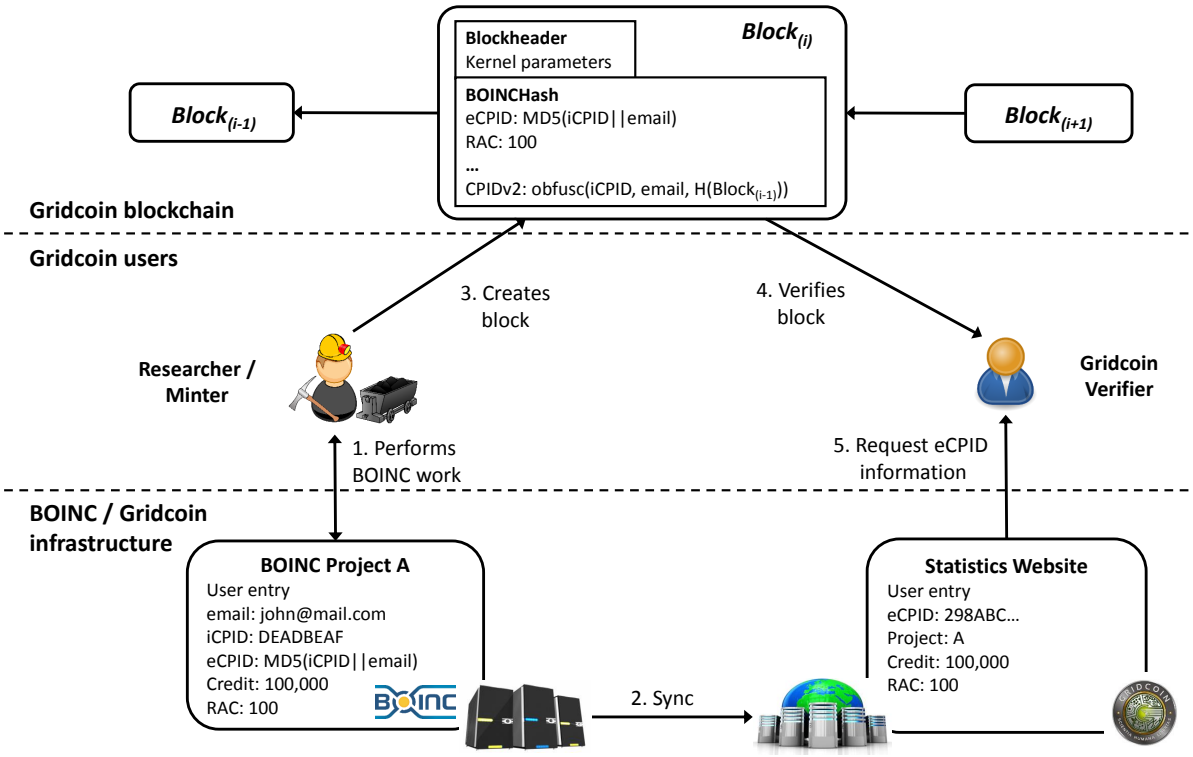

Figure 3: Gridcoin-Research architecture

4.1 Statistics Websites

In Section 2 the core concept behind BOINC was described. One functionality is the creation of BOINC Credits for users, who perform computational work for the project server. This increases the competition between BOINC users and therefore has a positive effect on the amount of computational work users commit. Different websites[4] collect credit information of BOINC users from known project servers and present them online. The Gridcoin client compares the RAC and total credit values stored in a new minted block with the values stored on *cpid.gridcoin.us:5000/get_user.php?cpid=eCPID* where eCPID is the actual value. If there are differences, the client declines the block. In short, statistics websites are used as control instance for Gridcoin. It is obvious that *gridcoin.us* administrators are able to modify values of any user. Thus, they are able to manipulate the amount of Gridcoins a minter gets for his computational work. This is crucial for the trust level and undermines the general decentralized structure of a crypto currency.

4.2 Project Servers

Gridcoin utilizes BOINC projects to outsource meaningful computation tasks from the currency. For many known meaningful problems there exist project servers[5] that validate solutions submitted by users,[6] and decide how many credits the users receive for their solutions. Therefore, the project servers can indirectly control the amount of Gridcoins a minter gets for his minted block via the *total credit* value. As a result, a Gridcoin user also needs to trust the project administrators. This is very critical since there is no transparency in the credit system of project server.

Centralization. There are several obstacles, which makes it hard to simply decentralize project server. For once it would be necessary to have a authentic distributed database of all users, which guarantees that the entries are just changed by the owner of user account. For SETI@Home, one of the scientific experiments of UC Berkeley, this would require 1.683.717 entries[7] and 524 megabytes on a single computer (in case it is stored in XML). BOINC also keeps track of the computer solving scientific problems for the projects. SETI@Home currently has 4.157.578 entries[8]. This requires 4.5 gigabytes of disk space on one computer. All these data need to be

[4] `http://boincstats.com`, `boinc.netsoft-online.com`

[5] `https://gridcoin.us/Guides/whitelist.htm`

[6] `https://boinc.berkeley.edu/trac/wiki/ValidationSummary`

[7] https://setiathome.berkeley.edu/stats/user.gz

[8] https://setiathome.berkeley.edu/stats/host.gz

transferred to all peers. We are not counting the overhead for error detection and correction, in case one host goes offline. The bigger amount of space would be needed for the raw data every client processes. SETI@home distributes work units of 350 kbyte size to every volunteer computer [12], every result of the processing is roughly 1 kbyte in size. Per hour SETI@home distributes 79,254 tasks[9] (work units) to the volunteer computers. So for one day in total we get 1,902,096 work units. This requires 667.635 Gbyte space in the blockchain just for one day, in order to allow other users to verify the results. Therefore, it is not just simply possible to decentralize the project servers.

5 Attacker Model

In addition to the trust issues identified in the previous section, Gridcoin suffers from serious flaws, which allow the revelation of minter identities or even stealing coins. Our attacks do not rely on the Gridcoin trust issues and the attacker does not need to be in possession of specific server administrative rights. We assume the following two simple attackers with limited capability sets.

5.1 Blockchain Grabber

For our first attack we only need an attacker with the capability to download the Gridcoin blockchain from an Internet resource and run a program on the downloaded data.

5.2 Gridcoin Minter

The attacker in our second attack is able to execute every action that can be performed by a regular Gridcoin user. This model has less capabilities than the *Web Attacker* model defined by Barth et al. [8]. In short, the user can use the source code provided on Github,[10] modify it, and run the client. Due to Gridcoin's peer-to-peer architecture, every active user is also a provider of resources for the network, such as bandwidth (message transport) and storage (blockchain blocks). As a result, every Gridcoin user and our attacker has control over some data in the network and can modify them. Furthermore, it is not necessary to participate as a researcher to execute our second attack. The attack can even be performed by a Gridcoin investor.

[9] Average value from the last 13 entries of Resultsreceivedinlasthour @ http://setiathome.berkeley.edu:80/sah_status.html (hint wayback machine)

[10] https://github.com/gridcoin/Gridcoin-Research

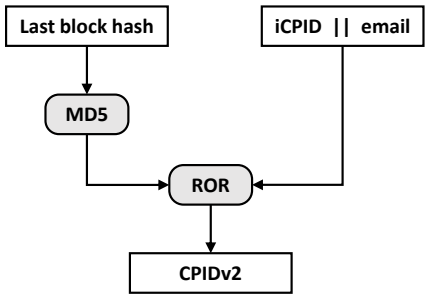

Figure 4: Obfuscation function for CPIDv2

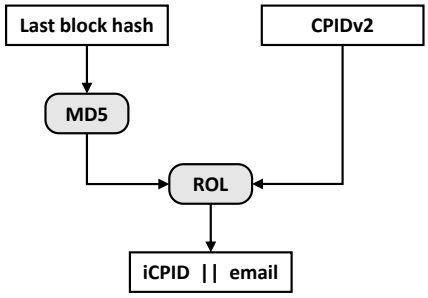

Figure 5: Deobfuscation function for CPIDv2

6 Privacy Revealing Attack

In the following we describe an attack on the privacy of Gridcoin, which allows an attacker to recover all email addresses of Gridcoin *Researchers*. We consider a simple attacker model with a blockchain grabber, who is able to download the Gridcoin blockchain.

6.1 Attack Concept

In order to protect the email addresses of Gridcoin *Researchers*, neither BOINC project websites nor statistics websites directly include these privacy critical data. The statistics websites only include eCPID entries, which are used to reward Gridcoin *Researchers*. However, the email addresses are hidden inside the computation of the *BOINCHash* (see Figure 6). A *BOINCHash* is created every time a *Researcher* mints a new block, and includes a CPIDv2 value. The CPIDv2 value contains an obfuscated email address with iCPID and a hash over the previous blockchain block.

By collecting the blockchain data and reversing the obfuscation function (cf. Figure 4 and Listing 4), the attacker gets all email addresses and iCPIDs ever used by Gridcoin *Researchers*. See the reversed obfuscation function in Figure 5 and Listing 4. The ROR function used in these steps computes the sum of two byte arrays (see Listing 3). ROL reverses this computation in an obvious way.

6.2 Evaluation

We implemented a deobfuscation function (Listing 4) and executed it on the blockchain. This way, we were able to retrieve all (2709) BOINC email addresses and iCPIDs used by Gridcoin *Researchers*. This is a serious privacy issue and we address it in Section 8.

7 Reward Forging Attack

The previous attack through deobfuscation allows us to retrieve iCPID values and email addresses. Thus, we have all values needed to create a new legitimate eCPID. This is required because the CPIDv2 contains the last block hash and requires a recomputation for every new block it should be used in. We use this fact in the following attack and show how to steal the computational work from another legitimate Gridcoin *Researcher* by mining a new Gridcoin block with forged BOINC information. Throughout this section we assume the Gridcoin Minter attacker model, where the attacker has a valid Gridcoin account and can create new blocks. However, the attacker does not perform any BOINC work.

7.1 Attack Concept

As stated in Section 7 the pre-image of the eCPID is stored obfuscated in every Gridcoin block, which contains a Proof-of-Research reward. We gathered one pre-image from the minted blocks of our victim and deobfuscated it. Thus, we know the values of the iCPID and the email address of our victim. Subsequently, use the hash of the last block created by the network and use these three values to create a valid CPIDv2. Afterwards we constructed a new block. In the block we also store the current BOINC values of our victim, which we can gather from the statistics websites. The final block is afterwards sent into the Gridcoin network. In case all values are computed correctly by the attacker, the network will accept the block, and resulting in a higher reward for the attacker, consisting of Proof-of-Stake and Proof-of-Research reward.

7.2 Evaluation

In order to verify our attacks practically, we created two virtual machines (R and A), both running Ubuntu 14.04.3 LTS. The virtual machine *R* contained a legit BOINC and Gridcoin instance. It represented the setup of a normal Gridcoin *Researcher*. The second machine *A* contained a modified Gridcoin-Research client 3.5.6.8 version, which tried to steal the Proof-of-Research reward of virtual machine R. Thus, we did not steal reward of other legit users.

Block	Time	PoR	PoS
632862	2016-08-11 21:09:52	4,84	0
631002	2016-08-09 20:45:20	2,91	1,48
630000	2016-08-08 18:50:56	0,87	0
629677	2016-08-08 10:49:52	0,43	0,01
629521	2016-08-08 07:05:04	18,95	0,19
622658	2016-08-01 00:07:12	11	0,7
618716	2016-07-27 18:34:08	5,6	0,33
616844	2016-07-25 18:33:52	3,22	0,31
615810	2016-07-24 14:54:08	0,75	0
615570	2016-07-24 08:30:08	0,34	0
615461	2016-07-24 05:39:28	0,9	0
615189	2016-07-23 22:26:08	1,58	0,18
614715	2016-07-23 09:45:52	1,58	0,01
614249	2016-07-22 21:05:20	41,66	1,86
601680	2016-07-08 23:47:12	0,57	0
601504	2016-07-08 19:13:36	0,33	0,51
601408	2016-07-08 16:33:04	2,05	0
600769	2016-07-08 00:03:28	3,33	0
599717	2016-07-06 20:27:44	3,22	0
598773	2016-07-05 19:54:56	0,14	0,21

Table 1: Blocks minted with the victim's eCPID

The victim BOINC client was attached to the SETI@home project[11] with the eCPID *9f502770e61fc03d23d8e51adf7c6291*. The victim and the attacker were in possession of Gridcoins, enabling them to stake currency and to create new blocks. Initially both Gridcoin-Research clients retrieved the blockchain from other Gridcoin nodes in the Gridcoin network.

The Gridcoin attack client made it possible to specify the victim email address, iCPID and target project. All these values can be retrieved from the downloaded blockchain and our previous attack via the `reverseCPIDv2` function as shown in Listing 4.

The attack client read the iCPID and email address of the victim from a modified configuration file. All other values, for example, RAC or ResearchAge, were pulled from `http://cpid.gridcoin.us:5000/get_user.php?cpid=`. As soon as all values were received, the client attempted to create a new valid block.

Once a block has been created and confirmed, the attacker received the increased coin reward, with zero BOINC contribution done. The attack could only be detected by its victims, because an outside user did not know the legit Gridcoin addresses a *Researcher* uses.

All blocks created with our victim eCPID are shown in Table 1. Illegitimate blocks are highlighted. We were able to mint multiple illegitimate blocks, thus stealing Research Age from our victim machine *R*. All nine attacker blocks passed the Gridcoin block verification, were confirmed multiple times, and are part of the current Gridcoin blockchain. During our testing timespan of approximately three weeks the attacker machine was wrongfully rewarded with 72.4 Proof-of-Research generated Gridcoins, without any BOINC work. The results show that the attack is not only theoretically possible, but

[11] `http://setiathome.berkeley.edu`

also very practical, feasible and effective.

The attack results can be reproduced with our Gridcoin-Research-Attack client.[12]

On the impossibility to identify previous attacks. There are two possible values, which can be checked in a new block in order to identify previous attacks: the Gridcoin address and the eCPID. The Gridcoin address is changed every time a new private and public key pair is created. This is a normal process to ensure privacy of transactions. The eCPID is updated to a new value every time the user participates in a new BOINC project or adds a new computer (e.g., if the user adds SETI@Home to his projects). So a change in the Gridcoin address or eCPID is not necessarily an indicator for an attack against a certain user. Therefore, it is not possible to verify if previous attacks based on our reward stealing attack were driven against the Gridcoin network. It would be possible to rent many virtual machines all over the world and triangulate [30] the minter in the Gridcoin network. Then it would be possible to look if a specific eCPID was used from very different locations. This would be a hint that there is an attack ongoing.

8 Proposed Countermeasures

In order to fix the security issue, we found one solution which does not require any changes to the BOINC source code nor the infrastructure. It is sufficient to change some parts of the already existing Gridcoin Beacon system. Thus, our solution is backwards compatible.

The current Gridcoin client utilizes so called Beacons to register new eCPIDs and stores them as a transaction of 0.0001 Gridcoins in a Superblock, which is created every 24 hours. A Beacon encloses the user's personal eCPIDs, a corresponding unused (but irreversible) CPIDv2 and the wallet's main Gridcoin payment address. Once the Superblock is created the eCPIDs is bound to one Gridcoin payment address. During the block verification process this bond is unfortunately not checked. Further the existing Beacon system does not use any strong asymmetric cryptography to ensure authenticity and integrity of the broadcasted data.

We propose to extend the Beacon System with public key cryptography. In detail we suggest that a user binds his fresh public key PK_1 to a newly generated eCPID, by storing them together in a Superblock. An initial Beacon would therefore contain a hashed (e.g. SHA-256) eCPID, the public key, a Nonce and a cryptographic signature created with the corresponding secret key SK_1 of the public key. This allows only the owner of the secret key to create valid signatures over blocks created with his eCPID. Thus, an adversary first needs to forge a cryptographic signature before he can claim Proof-of-Research work of another Gridcoin user.

$$Nonce \xleftarrow{\$} \{0,1\}^{128} \tag{9}$$

$$Input := \mathsf{H}(eCPID) \mid\mid PK_1 \mid\mid Nonce \tag{10}$$

$$FirstBeacon := \mathsf{Sign}_{SK_1}(Input) \mid\mid Input \tag{11}$$

For verification purposes nodes fetch the corresponding latest public key from one of the Superblocks. Furthermore, this Beacon structure allows a user to replace his previous public key associated with his eCPID. This is realized by submitting a new Beacon, with a new public key PK_2 signed with his old secret key.

$$Nonce \xleftarrow{\$} \{0,1\}^{128} \tag{12}$$

$$Input := \mathsf{H}(eCPID) \mid\mid PK_2 \mid\mid Nonce \tag{13}$$

$$UpdateBeacon := \mathsf{Sign}_{SK_1}(Input) \mid\mid Input \tag{14}$$

As all Beacons in the chain are verifiable and the latest public key is always authentic. The *Nonce* provide freshness for the signature input. This should prevent replay attacks against the Beacon system.
Note, that the eCPID needs to be completely unknown to the network, when sending the initial Beacon, for this concept to work as intended. The hash function ensures, that the Beacon does not reveal the fresh eCPID. As a result, an attacker is unable to mint with a eCPID even if he was able to intercept an initial Beacon and replaced the public key and signature with his own parameters, beforehand. This solution does not require any changes in the BOINC source code or the project servers.

Sign block. In order to claim the Proof-of-Research reward for a newly created block, the Gridcoin minter computes a signature over the hash of the blockheader. Afterwards, he stores the resulting value at the end of the corresponding block in a new field. The private key used for the signature generation must correspond to the advertised public key by the user. It is important to note that the signature value is not part of the Merkle tree and thus does not change the blockheader. In the end the signature then can be verified by every other Gridcoin user via the advertised public key correspond to the eCPID of the Gridcoin minter.

9 Related Work

The problems of Gridcoins basis, Bitcoin, and its design have been extensively studied and analyzed by researchers from different fields. Security relevant attacks

[12] `http://bit.ly/Gridcoin-Attacks`

and design flaws were identified back in 2012 [7] and 2014 [11]. Further studies [9, 19] measure the energy footprint and the future viability of Bitcoin's Proof-of-Work design concept. This research outlines the boundaries of Bitcoin and confirms the need for different design principles.

Apart from Gridcoin-Research various other altcoins attempt to resolve Bitcoin's problems, while retaining its advantages. Today, however, only few of the implemented design concepts have been analyzed sufficiently. Only altcoins, which introduced relatively subtle changes, like Litecoin [16] are well understood today. Other cryptocurrencies with massive profound changes and their own Proof-of-Works replacement include Peercoin [15], Blackcoin [29], Primecoin [14], Filecoin [5], or Permacoin [6]. The risks and drawbacks of the named altcoins are uncharted and their analysis is thus part of possible future work.

10 Conclusions

In the last couple of years, we could observe a huge increase of Bitcoin popularity. This was accompanied by the establishment of new alternative cryptocurrencies. Maintainers of these cryptocurrencies attempt to lure new users to invest their money and become a part of the community. The cryptocurrencies then become a larger part of the payment system and have easily market shares of several millions of US dollars, even though their security is not well analyzed and they may be lacking basic security design principles.

One of these cryptocurrencies is Gridcoin. It attempts to gain new users with a promise of meaningful computation which helps to heal cancer or study global warming. This is indeed a good purpose in comparison to Bitcoin-like wasteful Proof-of-Work computations. However, as we showed in our paper, in its current state the Gridcoin design is completely broken and insecure, and allows an attacker to steal Gridcoins from benign users.

We first conclude that future security research should concentrate on the analysis of cryptocurrencies in general. Given the amount of money transferred in Bitcoin-like currencies, these would deserve throurough security studies and audits. Second, we believe that one of the main research goals should become the study of meaningful computational problems for cryptocurrencies, so that these problems are directly used in Proof-of-Work computations. This is very challenging since the problems used in a typical cryptocurrency have to posses three general properties: It has to be (1) hard to solve, (2) verifiable (3), and the amount of reward has to be easily calculable. Analysis of meaningful problems with these properties and design of a secure cryptocurrency should also be coped with in the future.

References

[1] http://primecoin.io/

[2] Boinc Stats: Team Gridcoin Details, http://boincstats.com/en/stats/-1/team/detail/118094994

[3] Gridcoin: Market Capitalization, https://coinmarketcap.com/currencies/gridcoin/

[4] Map of Coins: Bitcoin, http://mapofcoins.com/bitcoin, as of Wednesday 26th July, 2017

[5] Filecoin: A cryptocurrency operated file storage network (2014), http://filecoin.io/filecoin.pdf

[6] Andrew Miller, Ari Juels, E.S.B.P.J.K.: Permacoin: Repurposing bitcoin work for data preservation. In: Proceedings of the IEEE Symposium on Security and Privacy. IEEE (May 2014)

[7] Barber, S., Boyen, X., Shi, E., Uzun, E.: Bitter to better—how to make bitcoin a better currency. In: International Conference on Financial Cryptography and Data Security. pp. 399–414. Springer (2012)

[8] Barth, A., Jackson, C., Mitchell, J.C.: Robust defenses for cross-site request forgery. In: Proceedings of the 15th ACM conference on Computer and communications security. pp. 75–88. ACM (2008)

[9] Becker, J., Breuker, D., Heide, T., Holler, J., Rauer, H.P., Böhme, R.: Can we afford integrity by proof-of-work? scenarios inspired by the bitcoin currency. In: The Economics of Information Security and Privacy, pp. 135–156. Springer (2013)

[10] Bonneau, J., Miller, A., Clark, J., Naryanan, A., Kroll, J.A., Felten, E.W.: SoK: Bitcoin and second-generation cryptocurrencies. In: IEEE Security and Privacy 2015 (May 2015)

[11] Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: International Conference on Financial Cryptography and Data Security. pp. 436–454. Springer (2014)

[12] Gardner, M.K., Feng, W.c., Archuleta, J., Lin, H., Ma, X.: Parallel genomic sequence-searching on an ad-hoc grid: experiences, lessons learned, and implications. In: SC 2006 Conference, Proceedings of the ACM/IEEE. pp. 22–22. IEEE (2006)

[13] Halford, R.: Gridcoin: Crypto-Currency using Berkeley Open Infrastructure Network Computing Grid as a Proof Of Work. self-published paper, May 23 (2014)

[14] King, S.: Primecoin: Cryptocurrency with prime number proof-of-work. July 7th (2013)

[15] King, S., Nadal, S.: Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. self-published paper, August 19 (2012)

[16] Lee, C.: Litecoin (2011)

[17] Moore, T., Christin, N.: Beware the middleman: Empirical analysis of bitcoin-exchange risk. In: International Conference on Financial Cryptography and Data Security. pp. 25–33. Springer (2013)

[18] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Consulted 1(2012), 28 (2008)

[19] O'Dwyer, K.J., Malone, D.: Bitcoin mining and its energy footprint. In: Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014). 25th IET. pp. 280–285. IET (2013)

[20] Ries, C.B.: UML for BOINC: A Modelling Language Approach for the Development of Distributed Applications based on the Berkeley Open Infrastructure for Network Computing. Ph.D. thesis, Glynd^wr University (2013)

[21] Various: Boinc credit computation, http://boinc.berkeley.edu/wiki/computation_credit

[22] Various: Boinc icpid and ecpid creation policy, http://boinc.berkeley.edu/trac/wiki/CrossProjectUserId

[23] Various: Boincpapers, http://boinc.berkeley.edu/trac/wiki/BoincPapers

[24] Various: Gridcoin classic wiki, http://wiki.gridcoin.us/Gridcoin-Classic

[25] Various: Gridcoin wiki, http://wiki.gridcoin.us/Proof-of-Research#Proof-of-Research_Reward_Calculation

[26] Various authors: Bitcoin Glossary, https://bitcoin.org/en/glossary/51-percent-attack

[27] Various authors: BOINC Credit statistics web sites and services, https://boinc.berkeley.edu/trac/wiki/CreditStats

[28] Various authors: Computation of the Recent Average Credit, https://web.archive.org/web/20120418125739/http://www.boinc-wiki.info/Recent_Average_Credit

[29] Vasin, P.: Blackcoin's proof-of-stake protocol v2

[30] Wang, Y., Burgener, D., Flores, M., Kuzmanovic, A., Huang, C.: Towards street-level client-independent ip geolocation. In: NSDI. vol. 11, pp. 27–27 (2011)

[31] Wustrow, E., VanderSloot, B.: Ddoscoin: Cryptocurrency with a malicious proof-of-work. In: 10th USENIX Workshop on Offensive Technologies (WOOT 16). USENIX Association, Austin, TX (Aug 2016), https://www.usenix.org/conference/woot16/workshop-program/presentation/wustrow

A Appendix

A.1 Gridcoin Additional Parameters

Due to the page limitation and in order to give a comprehensive overview, some additional parameters used in Gridcoin are described here.

The following parameters are part of the *BOINCHash* but left blank in every Gridcoin block.

- pobdifficulty
- diffbytes
- enccpid
- encboincpublickey
- encaes
- nonce
- NetworkRAC
- Organization
- OrganizationKey
- NeuralHash
- superblock
- CurrentNeuralHash

A.2 Code

The following section provides a more detailed and code-based description of the CPIDv2 calculation and reversion process initially described in Section 6. Listing 1 shows functions taken from the Gridcoin-Research source code, which are relevant for the CPIDv2 calculation. Note that these functions are obfuscated in the official source code and a manual deobfuscation was required to follow the process. For comparison the original code is shown in Listing 2.[13] Gridcoin-Research computes the CPIDv2 from the user's email address, the iCPID and the previous blockhash. Initially the function *HashHex* computes *shash* as the MD5 digest of the passed *block_hash* and consecutively applies the *ROR* function to the concatenation of iCPID and email (*boinc_hash_new*). *ROR* computes the sum of the ASCII values of the characters at *iPos* in *boinc_hash_new* and

[13] https://github.com/gridcoin/Gridcoin-Research/blob/8ac1a8c7de9155b42bdef7852adae81700c42366/src/cpid.cpp

Field	Description	Public
cpid	BOINC eCPID	+
projectname	BOINC project name	+
rac	BOINC RAC	+
clientversion	Gridcoin-Research client version	
ResearchSubsidy	equals ResearchSubsidy2	+
LastPaymentTime	timestamp of last PoR generated block with same CPID	+
RSAWeight	estimates coin gain for next 14 days	+
cpidv2	CPIDv2	-
Magnitude	magnitude	+
GRCAddress	minters Gridcoin address	
lastblockhash	hash of last block	+
InterestSubsidy	Proof-of-Stake reward	+
ResearchSubsidy2	Proof-of-Research reward	+
ResearchAge	Research Age used	+
ResearchMagnitudeUnit	magnitude unit (coin creation rate)	+
ResearchAverageMagnitude	average magnitude (average boinc work done by the researcher)	+
LastPORBlockHash	hash of last Proof-of-Research generated block	+

The **Public** column indicates whether the field is available on BOINC statistics websites. A blank cell indicates that the value can be freely selected and thus not relevant for our attacks.

Figure 6: *BOINCHash* structure contains fields that are of attacker's interest. For example, the CPIDv2 field is not available in the official BOINC statistics websites, but is useful for reconstructing minters' email addresses.

shash and encodes the result in hexadecimal (see Listing 3).

```
string ComputeCPIDv2(string email, string bpk, uint256 blockhash)
{
  return CPID().CPID_V2(email, bpk, blockhash);
}

string CPID::CPID_V2(string email, string bpk, uint256 block_hash)
{
  string non_finalized = HashKey(email, bpk);
  string digest = Update6(non_finalized, block_hash);
  return digest;
}

string CPID::Update6(string non_finalized, uint256 block_hash)
{
  string boinc_hash_new=bpk1+email1;
  string shash = HashHex(block_hash);
  for (int i = 0; i < (int)boinc_hash_new.length(); i++)
  {
    non_finalized += ROR(shash, i, boinc_hash_new);
  }
  return non_finalized;
}
```

Listing 1: Deobfuscated CPIDv2 calculation

```
CPID::CPID(std::string text,int entropybit,uint256 hash_block){init();entropybit
++;update5(text,hash_block);finalize();}template<typename T>std::string
ByteToHex(T i){std::stringstream stream;stream<<std::setfill(
((char)(0xbac+70-0xbc2)))<<std::setw((0x1344+4775-0x25e9))<<std::hex<<i;return
stream.str();}std::string CPID::HashKey(std::string email1,std::string bpk1){
boost::algorithm::to_lower(bpk1);boost::algorithm::to_lower(email1);
boinc_hash_new=bpk1+email1;CPID c=CPID(boinc_hash_new);std::string non_finalized
="";non_finalized=c.hexdigest();return non_finalized;}int BitwiseCount(std::
string str,int pos){char ch;if(pos<(int)str.length()){ch=str.at(pos);int asc=(
int)ch;if(asc>(0x87c+6520-0x21c5)&&asc<(0x1597+4174-0x259e))asc=asc-
(0x4c5+8720-0x26a6);return asc;}return(0x8b0+1872-0xfff);}std::string HashHex(
uint256 blockhash){CPID c2=CPID(blockhash.GetHex());std::string shash=c2.
hexdigest();return shash;}std::string ROR(std::string blockhash,int iPos,std::
string hash){if(iPos<=(int)hash.length()-(0x1f5b+1342-0x2498)){int asc1=(int)
hash.at(iPos);int rorcount=BitwiseCount(blockhash,iPos);std::string hex=
ByteToHex(asc1+rorcount);return hex;}return"\x30\x30";}std::string CPID::CPID_V2
(std::string email1,std::string bpk1,uint256 block_hash){std::string
non_finalized=HashKey(email1,bpk1);std::string digest=Update6(non_finalized,
block_hash);
return digest;}
void CPID::init(){finalized=false;count[(0x88d+1394-0xdff)]=(0x1fe5+1717-0x269a)
;count[(0x373+6812-0x1e0e)]=(0x65b+2790-0x1141);
state[(0xc88+3077-0x188d)]=1732584193;state[(0x1230+1876-0x1983)]=4023233417;
state[(0xada+3060-0x16cc)]=2562383102;state[(0x8d+3707-0xf05)]=271733878;}

void CPID::decode(uint4 output[],const uint1 input[],size_type len){for(unsigned
 int i=(0xbb+8818-0x232d),j=(0xcad+2297-0x15a6);j<len;i++,j+=
(0x150f+1818-0x1c25))output[i]=((uint4)input[j])|(((uint4)input[j+
(0x320+7218-0x1f51)])<<(0x1c36+2528-0x260e))|(((uint4)input[j+
(0x20c3+1239-0x2598)])<<(0x9d+5272-0x1525))|(((uint4)input[j+(0xbb7+2557-0x15b1)
])<<(0x131b+3426-0x2065));}

...
```

Listing 2: Original obfuscated CPID class responsible for basic Gridcoin calculations contains many magic numbers and is not formatted to hide the broken design

```
string ROR(string blockhash, int iPos, string hash)
{
  if (iPos <= (int)hash.length()-1)
  {
    int asc1 = (int)hash.at(iPos);
    int rorcount = BitwiseCount(blockhash, iPos);
    string hex = ByteToHex(asc1+rorcount);
    return hex;
  }
  return "00";
}
```

Listing 3: Deobfuscated ROR calculation

As *boinc_hash_new* is merely rotated a known amount of times, it is possible to reverse a CPIDv2, if the used previous blockhash is known. Our new function *reverseCPIDv2* shown in Listing 4 receives a CPIDv2 and a the used blockhash and calculates the respective email and iCPIDs. The underlying function *ROL* subtracts the corresponding ASCII value of *shash* from the supplied CPIDv2 and thereby reverses the function *ROR*, which is applied during the calculation process. More specifically equation 15 applies:

$$ROL(blockhash, iPos \cdot 2, ROR(blockhash, iPos, boinc\_hash\_new), iPos) = boinc\_hash\_new.at(iPos) \quad (15)$$

Hence our *reverseCPIDv2* function iterates over all relevant characters of the CPIDv2 to reveal the confidential user email and iCPID.

```
string ReverseCPIDv2(string longcpid,uint256 hash_block)
{
  string shash = HashHex(hash_block);
  int hexpos = 0;
  string non_finalized = "";
  longcpid = longcpid.substr(32, longcpid.length()-31);

  for (int i1 = 0; i1 < (int)longcpid.length(); i1 = i1 + 2)
  {
    non_finalized += ROL(shash, i1, longcpid, hexpos);
    hexpos++;
  }
  return (non_finalized);
}
```

Listing 4: CPIDv2 reversion

As the official source code inconveniently splits the calculation process into different functions we provide a simple and compact python implementation of the CPIDv2 calculation and the corresponding reversion algorithm. The code is shown in Listings 5 and 6.

```
import sys
import hashlib

# params: icpid, email, sha256 hash of previous block
def main(argv):
    # icpid + email
    userdata = argv[0] + argv[1]
    blockhash = argv[2]

    # md5(blockhash)
    digest = hashlib.md5(blockhash.encode("ascii")).hexdigest()
    cpidv2 = ""
    for i, char in enumerate(userdata):
        toAdd = ord(digest[i]) if i < len(digest) else 1
        if 47 < toAdd < 71:
            toAdd -= 47
        cpidv2 += '{:02x}'.format(ord(char) + toAdd)
    print(cpidv2)
```

Listing 5: CPIDv2 calculation in python

```
import sys
import hashlib

# params: cpidv2, sha256 hash of respective previous block
def main(argv):
    cpidv2 = bytes.fromhex(argv[0])
    blockhash = argv[1]

    # md5(blockhash)
    digest = hashlib.md5(blockhash.encode("ascii")).hexdigest()
    userdata = ""
    for i, _ in enumerate(cpidv2):
        toSubtract = ord(digest[i]) if i < len(digest) else 1
        if 47 < toSubtract < 71:
            toSubtract -= 47
        userdata += chr(cpidv2[i] - toSubtract)
    print(userdata)
```

Listing 6: CPIDv2 reversion in python