Submission date	3 June 2018
Grant name	On-Chain Voting Protocol
Organization / person	TFA / LUCIAP / FACTOMATIC / MFW
Previous Protocol Grant nr	N/A
Protocol Grant nr	FACTOM-INITIAL-GRANT-TFA-LUCIAP-FACTOMATIC-MFW-001
Sponsor	Factoshi ANO (@AlexanderSupersloth)

1. Executive Summary

Voting systems encompass different types of balloting: referendum (yes/no), multi selection options, scoring/grading, and opinion poll (free text). This is very flexible and open ended for usage. Additionally, the vote administrator must be able to choose what person, group, or entity to include or not include in the vote through the use of a system with digital identities.

Most immediate applications that the Factom community can benefit from an on-chain voting system are:

- Factom governance
- Vote of guides
- Vote of grants
- Scoring of grants
- Promotion and Demotion of Authority Nodes
- Integration with Coinbase Transactions

Authority node operators TFA, LUCIAP and Factomatic, and MyFactomWallet have joined forces to propose developing the first on-chain voting system for the Factom protocol.

Cost Summary by Entity

Entity	Primary e-mail	Discord POC	Cost (USD)
TFA	**ben@factoid.org**	**@KiwiWithLaserEyes**	**$105,091.20**
TFA / Sub MFW		**@DavidK**	**$ 43,200.00**
TFA / Material		**@bunfield**	**$6,000.00**
Factomatic	**vganev@factomatic.io**	**@sanchopansa**	**$45,400.00**
LUCIAP	**contact@luciap.ca**	**@luap**	**$24,200.00**
Factoshi (Sponsor)		**@AlexanderSupersloth**	**$3,500.00**
		Total	**$227,391.20**

2. Type of grant

[x] Software development [ ] Legal [ ] other

3. Project description

Section 2.5 of the Factom Governance Document requires voting in of guides with digital identities on protocol. Furthermore, the governance document also requires standing parties to be able to vote on grant proposals and other referendums as outlined in Section 4.3. The problem that needs to be solved is the development of a distributed, immutable, auditable voting system with un-censorable results using the Factom protocol. Additionally, the voting system must be able to to select what and who to include in the vote.

In this grant request we are proposing fully defining the specification for an on-chain voting system, developing a basic voting system to satisfy the immediate needs of the community (such as simple yes/no votes and multi-selection ballots), and finally researching and defining more elaborate voting schemes using technologies such as zero knowledge proofs (ZKP) and homomorphic encryption. Because of the significant unknowns about ZKP as it pertains to the Factom protocol, this grant only covers the effort of performing an analysis of ZKP/homomorphic encryption to define a scope of work for future development. If our analysis determines promising results for adding value to the protocol, our team will seek a new grant for the implementation.

4. Problem statement

For the purposes of this project, the team will be focusing specifically on building a sound infrastructure that can be expanded under a separate grant at a later date to support more complex automated functionality such as promotion and demotion of authority nodes and integration with coinbase transactions. For now, this project will provide the community with the base functionality for a voting system as well as defining a roadmap for follow-on development.

4.1 Background

There are several existing tools under active development that can be leveraged for this effort that were spearheaded by the team members who are part of this proposal. These development efforts include MyFactomWallet, Factom support for the Ledger Nano S, the factom.js JavaScript library, and the TFA Factom Block Explorer. Further efforts were put forth as well, to lay out a preliminary specification of a Factom on-chain voting system.

4.1.1 MyFactomWallet

MyFactomWallet's (MFW) mission is to provide a convenient and secure way to interact with the Factom blockchain. Factom aims to be a global utility for data integrity, and MFW was built to increase that global reach by removing some of technical barriers that exist today. MFW is an open source web application built with React and custom Factom Javascript libraries. At its core, it allows users to create new Factoid and Entry Credit addresses, and send transactions. The primary alternatives that exist today require users to either download the Enterprise Wallet desktop application, or to have the technical skills required to use a command line interface. MFW is committed to Factom's long term adoption by making it convenient for anyone to access the blockchain.

Security is a foremost concern for MFW and our users. MFW does not require user accounts and does not store your private keys. Though the site will be available online, users can also download the source code directly and run the application in an offline, air-gapped computer, to maximize security when generating new keys or signing transactions. MFW will also support hardware wallets, starting out with Ledger Nano S devices. This can keep user's keys safe when accessing the website through an unsecure device. Courtesy nodes are available, but users can connect to a custom node. These options allow users to make their own choices when balancing convenience and security.

MFW is designed to be useful for an array of stakeholders in the Factom Protocol. Investors will be more willing to hold Factoids when they can get them off exchanges, and take ownership of their private keys. A Factom Testnet courtesy node will be available so developers can more easily manage their Testoids and Test Credits. Factom evangelists will have an accessible web interface to assist in teaching new community members about the relationship between Factoids, Entry Credits, and the Factom Protocol. Finally, our close relationship with Authority Node Operators will help us determine our future priorities based on where we can

provide the most value, such building the on-chain voting interface as described in this document.

4.1.2 Hardware Wallets

As part of the initial MFW development, there was a desire to integrate the site with a hardware wallet. At the time, we were evaluating between the Ledger Nano S and Trezor wallets. The Nano S was selected as the first development target for integrating support for Factom tokens. The address schemes were developed first and successfully tested around Q4 2017 with a transfer of 1 FCT to the ledger FCT address. Transaction signing proved to be more of a challenge, however the first ledger-to-ledger transaction was successfully completed mid-March 2018. Since that time, Ledger has upgraded their firmware to close some security issues. The Factom wallet has been ported to the new firmware, however there is still testing and some integration with the middleware layer that need to be finalized. This middleware layer is a javascript interface that talks to the Nano S device. This middleware layer was recently updated to Node 9.5 to support the new API from Ledger.

4.1.3 Factom.js

Factom.js is the JavaScript library of reference to interact with Factom blockchain. It is available as a Node.js module and as a bundled file to be embedded in a web browser: this allows the library to be the foundational layer for a wide range of applications, from a Node.js backend service to a client side application. It is intended to remove the complexity of interacting with the raw factomd/walletd APIs, and it offers straightforward functionalities such as creating a chain, adding an entry, sending factoids, buying entry credits, creating multi input/output transactions, etc. While the library offers those high level functionalities for ease of development (similar to what factom-cli can offer), it still gives developers access to the full power of Factom technology. The library also takes care of limiting potential human errors and also to make all operations performant and reliable in a transparent way for the end user.

The library is currently in version 0.2 and has still plenty of development ahead. MFW is currently transitioning to Factom.js, and it is actively being used on several other interesting projects:

1. Ledger integration,
2. https://github.com/Factoshi/factom-accounts,
3. https://github.com/DBGrow/factom-dapps,
4. https://github.com/PaulBernier/factom-identity-cli,
5. https://github.com/PaulBernier/factom-storage
6. MyFactomWallet (under development)

4.1.4 Digital Identity Chains

Digital identity chains are the various chains on factom that can establish an 'entity' on factom. This entity could be a person, a company, or a group. The identity chain spec is described here:
https://github.com/FactomProject/FactomDocs/blob/master/Identity.md.

It is used for all authority servers, and can also be leveraged for a voting platform. Non-authority servers can follow this spec for a standard for creating and maintaining an identity. The spec can also be extended for application specific purposes, while leaving the base identity entries for the protocol to interpret.

4.1.5 TFA Factom Block Explorer

Blockchain explorers are crucial for any successful blockchain protocol, as it allows for an additional view into the blockchain for all the involved parties. It is used for development and information relay across the web such as transaction confirmation, block height and more. Data is what creates a blockchain, but it is not the only thing of interest. The current Factom explorer covers the data aspect; TFA's explorer is expanding beyond this.
In addition to the features of the current explorer, TFA's explorer will contain:

- Blockchain statistics such as those found here:
  https://blockchain.info/charts
- Identity Information
- Ability for node operators to verify successful identity creation before campaigning
- Visibility of all registered identities (Authority Set & others)
- Authority Information
- Keeping records of historical Authority Set performance
- Realtime consensus
- Monitoring Authority Set health from various data points to prove performance compliance
- Ability for Authority Set operators to track their performance
- Extendable to offer alert services

4.1.6 Zero-Knowledge Proof Systems and homomorphic encryption

Zero-knowledge Proofs (ZKP) is an iterative cryptographic protocol with two participants: a prover and a verifier. The protocol is used to demonstrate the prover's knowledge of something to the verifier (e.g. possession of a private key, solution to an equation, etc.) without revealing it. ZKPs are still an active research area, but are very powerful -- any problem in PSPACE can be cast in this framework. As part of this proposal, we will evaluate their suitability for the voting application and produce a detailed report with our findings, including a specification for a voting protocol which preserves the voter's anonymity.

4.1.6.1 Hidden What / Hidden Who

There are two main components of the voting system that can be anonymized: who cast a given vote, and which option was selected. We refer to them as the hidden what/hidden who. One approach for hiding the identity of a voter is the so-called **shadow identity generation**. Each voter would randomly generate a shadow personality, and then use a ZKP to make the following guarantees:

1. the shadow personality was linked to an authorized voter
2. each authorized voter could only create one shadow personality
3. nobody but the relevant authorized voter knows the true identity of the shadow personality

The ZKP ensures a link between a voter's public key and his shadow's public key. More specifically, it ensures that there's a number which can be used to efficiently and deterministically compute two public keys. The first key should be listed in the public set of voters, and the second one should be equal to the shadow's public key.

In addition to hiding the identity of a voter, it is also possible to hide the option they voted for. One way of accomplishing this is by means of homomorphic encryption, which together with ZKP can be used for hiding the votes in a simple yes/no vote or more elaborate techniques as described here.

4.2 Technical Approach

Section 4.1 describes the tools that are already developed or under active development that can be leveraged in support of this effort. As part of defining an approach for the voting system, there was some preliminary effort to outline a draft

specification of a voting system that leverages the Factom blockchain. In addition this can be integrated with existing wallets in development, namely the Ledger Nano S and MFW. Finally, cutting edge cryptography such as zero knowledge proofs can be explored as a potential solution for an on-chain voting system within the Factom protocol.

4.2.1 On-Chain voting protocol specification overview

In the currently suggested voting protocol, a vote administrator initiates the voting procedure by creating an off-chain description of the vote topic and the parameters associated with the vote, such as: voting options, timeline for submitting votes, deadline for publishing the results of the vote, terms for invalidating the vote, requirements for winning the vote, etc. A hash of the voting proposal is inserted into a specially designated Factom chain -- which ensures the terms of the votes are fixed and publicly auditable -- and the plain text of the proposal is broadcast to the participants via a separate channel (e.g. Discord, email, etc.). As part of the proposal, the vote administrator also publishes a set of digital identities, which are authorised to participate in the vote. Those are recorded in a separate Factom chain, referenced in the proposal.

The voting procedure is split into two phases: commit and reveal. In the commit phase, voters record an encrypted message of their vote on-chain. After expiration of this stage, they reveal the secret used to encode their vote, effectively making the option they chose public. Having a two-staged procedure ensures fairness of the vote, as no participant has knowledge of the running tally when they commit their vote. This eliminates psychological effects on voters, as they cannot be swayed by the current results. In practice, the two stages can be implemented using a **hash-based message authentication code (HMAC)**, signed by each voter**.** E.g., the commit entry can be of the form:

```
{
  "voterId": public identity (key) of the voter; must have a matching entry
         signed by the vote administrator in the vote-participants-chain to
         be included in the final tally,
  "voteHMAC": HMAC of the option that was chosen by the user,
  "signature": signature of the vote HMAC, created by the voter using a
         private key corresponding to the voterId
}
```

The corresponding reveal entry would then be:

```
{
  "voterId": public identity (key) of the voter (matching the ID in the
commit),
  "vote": unique identifier of the option that was chosen by the user,
  "secret": the secret used to generate the HMAC of the vote,
```

```
  "hmacAlgo": the hash function used to generate the committed HMAC (e.g.
        md5, sha1, sha256, sha512, etc.),
  "signature": signature of the vote HMAC, created by the voter using a
        private key corresponding to the voterId
}
```

Tallying of the votes would then be a straightforward aggregation of the vote choices in matching commit & reveal entries submitted by authorized voters.

For a more detailed description of the currently suggested approach, please refer to this specification.

4.2.2 Application Programming Interface (API) considerations

We will develop a core module in JavaScript that implements the primitives of the voting system. That core module will be the cornerstone of the interfaces we'll build on top: MFW web UI, Ledger integration, command line tool, APIs, etc.

Those primitives should allow the fundamental actions of:

- Creating a vote
- Participating in a vote (commit and reveal)
- Parsing a vote
- Validating a vote
- Tallying a vote

4.2.3 Extending client side voting within wallet

MFW will be extended with a voting section of the website. We will utilize the publicly available voting API and allow users to vote, create polls, and view results. Once the on-chain voting protocol research phase is complete, we will begin developing the UI in MFW. We believe providing a web interface for the voting process will increase participation and ease of access. We want to make it easy for users to test administering their own polls so they can become more familiar with how the protocol works. Users will be able to authenticate their actions using their digital identities.

4.2.4 Ledger Nano S and Factom Identity Support Integration

As part of this effort, EC support will be added to the Ledger Nano S. Furthermore, the Ledger Nano S firmware will be finalized and delivered to Ledger (the company) for integration into their ecosystem. Because the Ledger firmware is command-line

driven we will also support integration of the hardware wallet into the MFW client UI as part of this effort.

Finally, in an effort to link the Ledger in with the voting system, an evaluation will be performed as to the feasibility of integrating Factom identities onto the Ledger Nano S to support the voting system identities. One challenge with this approach is the Factom identities are not seeded and currently have no support for the BIP44 specification, both of which are the basis for the Ledger Nano S ED25519 implementation. We will evaluate multiple approaches to test for viability of integration of Factom identities.

4.2.5 Verification, Testing, and Acceptance

Every component will be tested according to software engineering best practices. Backend/API and command line tools will have automated unit tests and integration tests with a minimum of 80% line coverage. UI components should also have automated tests. All code will be peer reviewed and eventually open sourced.

Testnet will also be leveraged. The community will be engaged in a series of test votes as realistic as possible, and all functionality provided by the voting tools will be tested by the community. Testing in realistic conditions with external users will allow us to uncover potential bugs (users will be encouraged to try to break the application), and above all, test the user experience of the vote. We will gather the feedback from users in the early stages of development, as usability is an important aspect of the project.

Finally, we will define a series of acceptance test procedures that will accompany each development item.

5. Goals & Objectives

For the API development we propose the development of a "core" implementation in JavaScript that can be used by MFW, a command line tool, or a backend service providing API. That would be the cornerstone of the system. This will allow the tools to be built around this core, resulting in several primary deliverables. These would be:

- Core JS implementation
- MFW UI
- Ledger integration with MFW
- A cli implementation (standalone/command line tool)

In addition to the above, as part of this grant we will produce a detailed report with our findings on zero-knowledge proofs and homomorphic encryption, including a specification for a voting protocol, which preserves the voter's anonymity.

The development of these items can be parallelized and milestones derived. The objectives for this are divided into sub-tasks. These tasks will be focused amongst the team in accordance to the areas specialization. However, all team members will be available to assist in any tasking area as needed.

TFA

TFA's primary responsibilities will be to provide enhancements to the TFA Factom Explorer to provide backend certification for the vote system. TFA will further support the development of stand-alone client side tools to perform a similar function so that votes may be independently validated. TFA will finalize upgrades of the Ledger Nano S for Factom wallet support as well as support integration of the wallet into the MFW site. Furthermore, TFA will provide blockchain subject matter expert support for the the ZKP algorithm implementation on Factom.

FACTOMATIC

Factomatic will be responsible for researching the applicability of zero-knowledge proofs and homomorphic encryption to the the voting protocol and for the production of a report, which will summarise those findings. The report will include a detailed specification for a voting protocol, which preserves the user's anonymity.

In addition to this, Factomatic will support the development of the initial voting systems by contributing to the factom.js library, doing code reviews and performing tests of the system.

LUCIAP

LUCIAP will take part in the design discussions, review and sign off of the protocol specification. LUCIAP will be leading the core JavaScript implementation of the voting protocol (especially as it will be built on top of factom.js, library developed by us). LUCIAP will also participate in the development of the stand-alone application

(command line tool). Finally LUCIAP will be involved in all the tests/validations aspects of the project.

MFW

MFW will be responsible for the client-side wallet UI as part of this effort. This UI will include integration with the Ledger hardware wallet, as well as adding extensions for the on-chain voting interface and integrating with the newly developed JavaScript voting API. MFW will additionally be involved with reviewing and testing the voting protocol.

Project Tasks

Task 1: System Design

The team will design a voting structure under Task 1. This includes the development of cryptographic algorithms that satisfy protocol voting requirements outlined above. This task is divided into the following sub-tasks:

Task 1.1: On-Chain Vote Specification Refinement: For this task the team will work with Factom Inc. to refine the voting protocol specification identified in section 4.2.1.

Task 1.2: Release an official document of the voting protocol specification: This task will involve releasing the voting protocol specification to the community and vested parties for review. Feedback will be considered and voting specification will be refined and finalized.

Task 1.3: Wallet interface design: For this task, the team will design the UI interfaces needed to support the interactions of the voting system with the user.

Task 1.4: Wallet library design: For this task, the team will design the library interfaces to connect the UI with the blockchain for the voting specification.

Task 1.5: Hardware wallet investigation: This task will involve research into the feasibility of the Ledger Wallet supporting Factom Identities.

Task 1.6: Voting verification system: For this task, the team will design the backend system needed to efficiently verify votes.

Task 2: Development

Task 2.1: Core JS implementation: This task will involve JavaScript implementation of the voting specification resulting from Task 1.1.

Task 2.2: Complete wallet UI in MFW: For this task, the wallet UI will be completed to provide full client-side functionality

Task 2.3: Voting UI in MFW: This task involves developing and integrating the voting API's into the UI for the voting system.

Task 2.4: Ledger EC support and middleware updates: This task will be to

add EC support and finalize the Ledger middleware layers needed to interface with MFW.

Task 2.5: Ledger integration: This task will integrate the hardware wallet functionality into the MFW UI.

Task 2.6: Command line tool: As part of this task an independent vote verification command line tool will be developed and integrated with the Core JS api.

Task 2.7: TFA Factom Explorer extension: This task will involve developing the extension to the TFA explorer that will provide real-time feedback and verification of voting results.

Task 3: Testnet Deployment and Acceptance Test Plan Development (ATP)

The team will develop a verification and acceptance test plan for the on-chain voting system, MFW and its supporting tools.

Task 3.1: Define testing methods: For this task, the team will define testing methods for the voting system.

Task 3.2: Compile an Acceptance Test Plan: Under this task, the team will develop an acceptance test plan in accordance to the development items defined under task 2.

Task 3.3: Develop and deploy on testnet: For this task, the team will develop test cases and deploy on the testnet.

Task 3.4: Perform acceptance testing dry run on testnet: For this task, the team will execute the ATP on the testnet and perform corrective actions identified by the outcome of the preliminary acceptance testing.

Task 3.5: Deploy on mainnet: For this task, the team will develop test cases and deploy on the testnet.

Task 3.6: Perform formal acceptance testing on mainnet: For this task, the team will perform a formal acceptance test with the Sponsor. The results of this test will be provided to the community and incorporated into the final report.

Task 4: Zero-Knowledge Proof Assessment for the Voting Protocol

The team will perform an assessment of ZKP within Factom. This task is broken down into the following subtasks:

Task 4.1: Evaluate use of the Hidden What / Hidden who: The team will study existing techniques of voting systems used on other onchain voting systems that are designed around ZKP.

Task 4.2 Evaluate the use of homomorphic encryption: For this task, the team will perform an feasibility assessment of using homomorphic encryption within the on-chain voting system.

Task 4.3: Prepare report and define level of effort for ZKP integration: For this task, the team will use the feasibility study to develop the requirements

and define the scope of work for ZKP integration into the voting specification for a follow-on effort.

Task 5: Final Report

Upon completion of Tasks 1-5, the team will prepare and deliver a final report. This report will include the design specifications and documentation for the on-chain voting system. This final report will also include the analysis of the ZKP/homomorphic encryption as well as recommendations for a path forward into future development. This could serve as a basis for a new grant for the actual implementation.

6. Success criteria (measurable)

Criteria	Measurement
Voting Specification Released to Public for review	Inspection and Analysis
Nano S Factom Wallet Accepted by Ledger	Inspection
MFW Site Launched and passes acceptance test on mainnet	Demonstration and Test
Sample vote on test net	Demonstration

7. Timeline, activities & milestones

Activity	Milestone	Timeline
Voting Specification Released for Review	1	Week 3
Ledger Factom App released to Ledger (Company)	2	Week 5

Voting Specification Finalized	3	Week 5
MFW beta released on test net	4	Week 8
Voting System Released to test net	5	Week 8
Acceptance Testing on mainnet	6	Week 10
Final Report	7	Week 12

8. Budget(s)

PROJECT: ON-CHAIN VOTING PROTOCOL COST PROPOSAL

GRANT NUMBER: FACTOM-INITIAL-GRANT-TFA-LUCIAP-FACTOMATIC-MFW-001

Milestone	Description	Funds Requested
1	Voting Specification Released for Review	$ 24,732.80
2	Ledger Factom App released to Ledger (Company)	$ 17,808.00
3	Voting Specification Finalized	$ 6,697.60
4	MFW beta released on test net	$ 40,089.60
5	Voting System Released to test net	$ 49,000.00
6	Acceptance Testing on mainnet	$ 30,211.20
7	Final Report	$ 55,352.00
	Factoshi (Sponsor)	$ 3,500.00
Total		$ 227,391.20

Additional information on cost proposal can be found here. Work will begin one week after award unless otherwise coordinated with our sponsor. Note: Milestone 2 includes a 5000 euro listing fee for the ledger app.

9. Competition & collaboration

We have formed a strong team of collaboration between three Authority Node Operators and MyFactomWallet. Each team organization brings a unique capability that will help ensure the successful execution of this Grant.

LUCIAP

Luciap is a Federated servers operator company that is also developing factom.js, the JavaScript library of reference to interact with Factom blockchain. Thanks to the development of this library we gathered important technical knowledge of Factom mechanisms and APIs. We were among the reviewers of the initial voting protocol spec. The factom.js lib offers a version that can be embedded in web browsers allowing to perform Factom related operations on the client side without having to communicate critical data (secret keys) to the server: this is an important requirement for the voting application.

The Factoid Authority (TFA)

TFA is an Authority Node operator with expertise that lies mainly in software development and project management. TFA is producing a Discord Monitoring Bot for Authority Nodes, which notifies Operators of server-related issues; it contacts members based on expertise & time-zone. We will be open-sourcing this soon for the benefit of all Operators. TFA is also working on a new Explorer, with added features. Eventually it will include Identity Information, Authority Information & Real-time Consensus.

MFW

MFW is a team of passionate developers who have been building the MFW UI, JavaScript libraries, and Ledger Integration needed to bring a convenient and secure web wallet to Factom. Many members of the MFW team have become Authority Node operators, giving us special insight into the needs of the evolving Factom ecosystem.

FACTOMATIC

Factomatic LLC is an Authority Node Operator company that also provides consultancy services to new and existing businesses interested in using the Factom protocol. We are the main authors of the suggested on-chain voting protocol, reviewed by multiple members of the community. In the last weeks, we have been investigating the possibility of extending the voting protocol using zero-knowledge proofs, which would resolve the main problem with the currently suggested approach: voting being non-anonymous.

10. Organization or Person info

Key personnel from each organization has been identified to support this effort.

Steven Masley (TFA, @SteveM)
Steven is a software developer and student. He has served as an independent contractor for Factom Inc. since May 2016 and is currently a senior blockchain developer for TFA. He has extensive experience developing applications on top of the Factom blockchain, with notable accomplishments as a Factom core developer which includes implementing the identity system currently used in factomd. Mr Masley also served as the Project Manager for the MyFactomWallet development and continues his efforts as a Factom core developer. He is continuing his studies at the Rochester Institute of Technology pursuing a BS in Computer Science with expected graduation in 2019.

Valentin Ganev (FACTOMATIC, @sanchopansa)
Valentin has over 12 years of experience in tech startups, acting as an architect and a core developer of state-of-the-art solutions in cloud computing, big data analytics and distributed ledger technology. He has extensive knowledge of Linux and software engineering and is proficient in Python, Scala and the Spark data analytics stack. Valentin has been involved in projects ranging from real-time monitoring tools for cloud computing platforms in the US, through government-funded smart cities capabilities development in Singapore to real-time traffic jam detection on Swiss highways using telco data. Valentin is the main author of the currently suggested voting protocol. He holds a MSc in Robotics from ETH Zurich.

Paul Bernier (LUCIAP, @luap)
Mr. Bernier has worked as software developer at Amazon for over 3 years and is certified AWS Solution Architect associate. He has professional experience in developing software in JavaScript (Node.js) and architecting reliable and performant systems. He also has a background in security as he was awarded the title of ESSI (Security Expert in Information Systems) by the ANSSI (National Cybersecurity Agency of France).

David Kuiper (MFW, @DavidK)
David Kuiper has worked as a full-stack web application developer for the past 5 years. He recently served as development lead for a small team tasked with designing and integrating multiple enhancements into a complex budget tracking

tool used by the U.S. Veterans Association. He has professional experience in the full range of implementing a software release, including requirements gathering, impact analysis, design, development, testing and deployment. He holds a BSc degree in Computer Science from James Madison University.

Dennis Bunfield (TFA, @bunfield)
Mr. Bunfield has over 25 years of professional software development for high performance real-time hardware-in-the-loop simulations. For the past 10 years, Mr Bunfield has managed a team of 11 physicists and software engineers while managing contracts with budgets in excess of $3M/year. He has also developed support for FCT on Ledger Nano S hardware wallet which is currently in testing. He holds a MS in Physics from the University of Tennessee Space Institute.

11. Consultant(s) and Sponsor

The Factoshi Authority Node Operator will be used as the sponsor of this Grant effort. They will provide support to the team in the form of ensuring deliverables are received, hours / costs on the project are being tracked in accordance with best contract accounting practices, will support witnessing and signing off on acceptance test(s) for each milestone of this effort, and relay progress reports to the community from the team.

In addition to the sponsor, Factom, Inc will be used as a consultant on this project. Factom will provide this team with support through funding funding from the Factom Developer Support Grant. The focus of their support will be in the design of the data structures within the voting system to ensure all voting on-chain system development requirements are captured.

12. Additional documentation

REFERENCES

https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell/

https://medium.com/@VitalikButerin/zk-snarks-under-the-hood-b33151a013f6

http://www0.cs.ucl.ac.uk/staff/J.Groth/ACNS05VoteProofFull.pdf

http://blog.stratumn.com/zkp-in-anonymous-publicly-verifiable-boardroom-elections/

https://courses.csail.mit.edu/6.857/2016/files/30.pdf

https://security.stackexchange.com/questions/35813/homomorphic-encryption-used-for-e-voting

13. Indemnification

By submitting a grant proposal or participating in the grant proposal process, the submitter indemnifies and holds harmless all Guides, Authority Set Members, and Standing Parties, as it strictly relates to the grant, against any loss or expense incurred, but which in no case will exceed the total dollar amount of the grant as issued, by accepting or conducting the work of a grant award by reason of the fact that the Guides, Authority Set Members, and Standing Parties including, without limitation, any judgment, settlement, attorneys' fees and other costs or expenses incurred in connection with the defense of any actual or threatened action or proceeding, provided the loss or expense resulted from Good Faith Errors or from action or inaction taken in good faith for a purpose which the Guides, Authority Set Members, or Standing Parties reasonably believed to be in, or not opposed to, the best interests of the Factom Protocol.

Note: Please see Governance for proper definitions of Guides, Authority Set Members, and Standing Parties. Grant proposals submitted in another format shall have this indemnification.