CROWN PAPER

2.0 INTRODUCTION & FEATURES

The Core Dev Team
December, 2016

"The first problem is that for most programs, there is no clear description of what they are supposed to do. If there are no specifications, then you cannot even check whether a program is correct or not. For such programs, the whole concept of correctness is undefined.

Many software projects have a document called the functional specification. In theory, this should be the specification of the program. But in practice, this document often does not exist, is incomplete, or specifies things that are irrelevant for the behavior of the program. Without clear specifications, there is no hope of getting a correct program."

- *Niels Ferguson, Brice Schneier, Tadayoshi Kohno.*
  Cryptography Engineering: Design Principles and Practical Applications

"One must think like a hero in order to behave like a merely decent human being"
- *May Sarton*

ALL BEGINNINGS ARE HARD

Crown did not spring into existence fully formed. Instead, as discussed in the first paper, the code is just emerging from the mud. The plan is for the Crown platform is to be nothing more than a minimalistic core which verifies and validates transactions and updates the *Chronicle* – a more intuitive name for the "blockchain" registering events and transactions on the Crown network. The Core does nothing else. It gets a transaction in the form of a message and says thumbs up or thumbs down. *All the platform will dictate is what goes in the Chronicle.*

The rest will be up to you, or others like you. The rest of the Crown Platform is driven by creating an open space in a crowded world. Then putting out a few tools and seeing what we will all create.

The opening quote provides context for what we are doing with this series of papers. The Crown Papers are being generated from the work documenting the Requirements and Functional Specifications for the core software. Our ultimate goal is to get what Ferguson, Schneier and Tadayoshi describe as a correct program. Or, applying May Sarton's perspective, "one must work like a hero in order to produce code that is merely correct."

The requirements portion of the documentation describes what a feature is intended to do, then the function spec describes how it should work in terms of user interaction, input and output. The final step is the implementation design and coding. This paper introduces a few of the tools outside of the Core which we are in the process of "spec'ing out" (writing the functional specifications for), and discuss conceptually how those tools might work.

NOT EVERYTHING IS FOREVER

In writing these papers we inevitably end up with different versions and drafts and iterations. Iteration and revision are normal -- not the exception, but the rule. We all know this is how not just writing, but much of life works. Why do we expect a transaction network and technology platform involving complex transactions and dependencies to be any different?

One requirement of the Crown platform is that we have a way to draft, test and propose transactions. These would be transactions which we are considering doing but which haven't been submitted to the core or the Chronicle yet. We would also like to be able to not just draft transactions, but actually test the result of a transaction script to make sure that it does what we would like it to. This is equivalent to stating that another requirement of the platform is that it supports the testing of transaction scripts, and can store transaction script templates.

If we step back for a moment, to a higher level of abstraction (when designing and coding software one is always mentally shifting between different levels of design perspective and checking ideas against a more granular level of implementation design), we realize that there is also something else going on. We are discussing a category of activities which involve the Crown platform, but are not immediately using the Chronicle or Core Wallet. These are activities which may reference the permanent record, but are not implementing transactions at this moment. The requirement that seems to fit this, would be that we have an application, which is a sort of Core Wallet plus other features whose characteristic is that everything being done in it is subject to change, or temporary.

We think of the process of creating these draft transactions as being like keeping an accounting journal, or journaling. The portion of the platform in which one would create draft transactions is the ***Journal***.

But draft transactions are just one of a host of activities which could be enabled by the public-private key infrastructure of the Chronicle. Applications, or other subscriptions, could also be verified through the Chronicle – but the use would need an interface for working with the application. The need for the interface to applications is another requirement of the Journal.

The Journal is the interface to any temporary use for the Crown Platform. That which is permanent or involves a commit of data or a transaction to the public permanent record stored in the Chronicle goes through the Core. Everything else interacts through the Journal.

The space for constant iteration and revision enabled by the Journal should also help catalyze innovation on the platform. The Journal is also required to be a testbed for new chains, or semi-private chains.

But it's not much good having different ways of doing things if the methods are all embedded in a console window somewhere and only known to those who designed it. For these features to be valuable, we need to make them usable and document what the features are and how they can be used.

The first white paper introduced the ***Lexicon*** as the dictionary of terms describing the platform, but another way to think of the *Lexicon is as the super-set of the documentation of the Crown Platform.* Just as the Lexicon is intended to use language and logic to make the Crown platform understandable, we plan to also have a way of interacting with the platform through a simple interface. Messaging apps have spawned the use of aliases and acronyms, and Crown will just build on what has happened organically with a set of defined phrases that can be used and that the Journal knows how to interpret or parse. This set of phrases and structures is ***Shorthand***.

Shorthand is will not support looping or make any attempt to be a Turing complete programming language. There are already some great programming languages and great people working on making them better. Shorthand's primary focus is to provide simpler interface on existing scripting, API and CLI

tools in the Bitcoin and Dash nodes and whatever minimal additional capabilities the Crown team may add– making them all easier to use.

Our underlying assumption is that before we need to add extensive functionality to the platform, we should expose some of the incredible tools which are already embedded in the Bitcoin and Dash cores.

The Journal, Lexicon and Shorthand are designed to be perpetually changing and evolving components of the system. Their purpose is to be where plans are drafted, applications accessed and new combinations and permutations - new ideas can be played with and saved, but not necessarily committed to the Chronicle - yet.

COMBINATIONS AND PERMUTATIONS

"But this one goes to 11"

- *This is Spinal Tap*

The math behind combinations and permutations is one of the wonders of the world, almost as amazing as compound interest and exponents. Since we are boiling down the core to its essential elements, those elements are really a set of functions involving a digital token, a time stamp, and a public key. So then logically the functions of the core are limited to the combinations and permutations of those three items. That's it. Of the three, we are not playing with time – not only is it beyond our powers, but the purpose of the whole system is to assign a time order to transactions, so scrambling time would sort of sabotage that. Which means that in the core we can only play with types of tokens and keys. For now, we will assume that Crown is the only token we could work with (a completely unrealistic and also undesirable scenario, but work with me – we are early in the design process). So that means that the only item left to play with is the structure of the public keys used in signing transactions.

If having one public-private key pair creates all this possibility, one might reason that 2 or 3 or 4 would be better and better and better. We aren't sure if this is the case, but we do know that evolutionary algorithms use crossover and mutation and that there might be use cases where having a set of all purpose and changeable keys associated with a specific single purpose and immutable key could be useful.

We have some of these on a keychain in my house. We call them ***Skeleton Keys***. The idea is to associate a set of mutable public keys (*skeleton keys*) with an immutable public key (*token address*), and allow the construction of new transaction types to allow for the use and updating of the skeleton keys.

While the Skeleton keys are mutable, they are also part of a transaction – so they are part of the permanent record, but would only be active, well, after they have been activated. As Satoshi defined it, the Chronicle is a timestamp server after all.

The skeleton key requirement is that we be able to create a transaction type that includes a series of hashes, which may be similar to the Crown address, or may not. The hash method could be an option in creating the skeleton key and dependent on the purpose of the key. The other requirement for skeleton keys is that the implementation should be such that it helps minimize the wear to other keys.

Another way to think of this design feature would be that the Chronicle would be a public key infrastructure in which each immutable public key can be associated with up to "n" mutable skeleton keys, enabling signatures which could be scripted from the combination of a matrix of keys.

That's about all it makes sense to say about the skeleton keys right now. *Part of the power of a skeleton key is its mystery – you don't know which or how many doors it may unlock.*

DESIGN PRINCIPLES

The ultimate code and logic of the Crown platform will not be a mystery. In the first paper we laid out a series of values and goals, linked to a governance model and idea that the series of transactions aren't a chain, but a Chronicle.

In this paper we introduced some of the key design elements of the platform:

1. SIMPLE CORE
   Security comes from simplicity. Keep the core of the platform as simple as possible.

2. DRAFT MODE:
   Differentiate between the permanent and the "draft" or temporary modes.

3. USER FRIENDLY:
   Prioritize usability in every step of the design for the platform.

4. PLAYGROUND:
   Make space for play and creativity in the platform, it's where innovation will come from.

5. COMMUNITY:
   Open the design discussion to the community, the platform is theirs.

These design principles along with the values in the Knowledge, frame the requirements for the platform and provide a language for discussion and debate as the platform evolves.