Pastel Whitepaper

Jeffrey Emanuel

September 13th, 2018

Contents

1 Introduction

We introduce **Pastel**, an open-source, decentralized system which allows artists to register "provably rare" digital artworks on a Bitcoin-like blockchain, while also allowing art collectors to purchase these artworks and "trustlessly" trade them among themselves without reliance on a central authority. The resulting rare digital artworks can be thought of as continuing the long tradition of limited edition art prints such as those sold by art galleries, often in sets of 10 individually numbered pieces. Unlike such "physical world" prints, rare dig-

ital artwork–when implemented correctly using a secure cryptographic digital signature scheme and a robust "near-duplicate" image detection scheme–offer the digital art collector a high degree of certainty in determining the authenticity and provenance of a specific artwork registered in the system. At the same time, rare digital artwork allows artists to directly access and trade with their patrons, without an art gallery or dealer taking a huge slice of the value. Furthermore, rare digital art collectors can trade among themselves using an integrated, decentralized art marketplace at a minimal cost in terms of fees, providing low-friction liquidity to a notoriously illiquid and fee-ridden traditional art market.

2 Overview

The purpose of this document is to provide a detailed description of the Pastel project, including:

- An introductory overview of the key ideas and functionality of the project (this section).
- A discussion of the market opportunity and economic rationale for the project.
- A step-by-step description of the process of registering a new artwork on the system and then viewing/trading the newly registered asset.
- A review of the theoretical ideas from computer science and cryptography used to build the system, and a discussion of how these concepts interface with important ideas from game theory and economics to allow us to construct secure decentralized systems with productive incentive structures that promote useful network behavior.
- A detailed analysis and explanation of the key components of the system, including the file storage layer, the blockchain ticketing system, and the various image processing functions required to regulate the content registered on the network, among others.
- A summary of the new technical contributions made by the project, as well as a discussion of future potential functionality as the project is refined and expanded over time.

We provide here a brief overview of the project, but each of the concepts and features mentioned below will be described in further detail in subsequent

sections of this document.

Intended Audience:
This Whitepaper attempts to strike a middle ground between being easily understandable by a conscientious (albeit less-technical) reader, while at the same time describing the system in sufficient clarity, detail, and rigor so that an expert can be reasonably convinced by the arguments and merit of the ideas presented. To fully appreciate the ideas contained below, the reader should be reasonably familiar with the design of *Bitcoin*, *Z-Cash*, and *Dash*, as described in the respective Whitepapers for each project. In addition, certain ideas relating to other aspects of the system, such as the machine learning and statistical tools used for image processing, or the use of distributed hash tables in the off-chain storage system, may be unfamiliar to readers. While we attempt to give sufficient background to understand the core ideas involved, the reader is encouraged to research these areas independently in order to better understand the design challenges faced and trade-offs taken in the project.

2.1 Existing Work

While the use of a blockchain or distributed ledger as a registry for artwork is not a novel concept (see the Existing Work section below), no existing project has solved the particular challenges that arise in developing an art registry system that can work reliably and securely in a truly decentralized way. Existing projects such as PepeCash/Rare Pepe, which briefly reached a valuation of $90 million in early 2017, established that there is market interest in rare digital artwork. Yet PepeCash is far from decentralized; although the details of the artworks and their ownership are stored on the Bitcoin blockchain (by means of the *CounterParty* protocol), the registration process to create a new artwork relies on contacting the creator of the project, Mike, who unilaterally decides whether to allow the registration using criteria such as the perceived uniqueness or originality of the submission (e.g., you can't register a "Simpsons" Pepe because someone already did that).

Furthermore, the purportedly decentralized trading of "Rare Pepe" artworks also relies on centralized servers to match orders and to offer "secure" escrow services, though, to the project's credit, two "independent" servers of this kind are offered to the public, and the code to run a server is available in an open-source code repository. In practice, however, such an approach means that the network and the services it provides (i.e., registering new rare digital artworks and trading them for coins), are inevitably brittle and easy to attack if the project became a valuable enough target. Not to mention that Rare

Pepe is also exclusively focused on "art" that pertains to the popular "Pepe the frog" character made famous in Internet memes, rather than open to digital art in general. There are several other blockchain projects, such as *Dada* and *Verisart*, that attempt to address similar problems, but we believe that all of these projects suffer from multiple, fatal flaws that make them wholly inappropriate for serving as a storehouse and registry for digital artworks.

In short, other existing "art on the blockchain" projects put far too low of an emphasis on decentralization and security. They attempt to solve problems through heavy-handed, blunt approaches that make them unattractive for serious artists and collectors, and generally introduce huge amounts of centralization through human-managed moderation that is subject to subversion. For example, Dada forces artists to use their proprietary, web-based software to create the images, which is little more than a primitive *Microsoft Paint* clone– and that is the *only* way to submit art to the project. Why should an artist who has spent decades mastering powerful tools such as *Adobe Illustrator* or *Photoshop* be hamstrung by such an arbitrary requirement?

If this weren't bad enough, Dada utilizes an outright socialistic redistribution policy: if an artists wants to sell their work on the Dada network, then a huge percentage of the sale proceeds (an insanely high 45%! [CHECK THIS]) is taken by the network in the form of a "tax," so that it can be distributed to other artists on the network. While a technical Whitepaper is perhaps not the forum for discussing the merits of socialism versus free-market capitalism, it seems like a horrendous idea to force successful artists, who can find a ready market for their works, to subsidize less talented and successful artists who can't convince collectors to willingly purchase their works.

Other systems, such as Rare Pepe, do not even attempt to store the underlying image files themselves in the network: these are instead hosted on a traditional, centrally controlled website. That is, the underlying digital artwork assets that people buy and sell don't even make reference to the underlying image file associated with the artwork, other than to perhaps reference a simple file hash. What if the owner of the centralized website stops paying the hosting bills? What would happen to all the artwork? Does this seem like a wise idea to have a system where the images– the actual artwork itself in the traditional sense of the word– is so vulnerable and centralized?

Even if these competing projects didn't suffer from such "show-stopping" flaws, they are inherently less ambitious and powerful than what we are attempting to do in the Pastel project: we are leveraging the latest advances in machine learning, as well as novel contributions from the Pastel project, to remove the human element from the equation to the greatest degree possible. We are building a system in which machines can coldly and objectively

evaluate which images should be prevented from being registered–say, because they contain child pornography, or are "rip offs" of existing registered artworks already in the network. We are building a system where anyone can, without permission or authorization, create original digital artworks using the tools of their choice, and submit these for registration without any individual person or entity being in a position to impede them. And we are building a system where the actual art files are stored in a robust, decentralized way, and where registered artworks can be freely traded in a completely trustless way. Perhaps most importantly, we are building a system in which anyone can help *run* the network themselves by purchasing a sufficient number of coins so that they can setup a machine as a so-called Masternode, described more fully below.

2.2 Project Guidelines

Instead of attempting to serve as a universal platform for building decentralized applications, the Pastel project focuses narrowly on digital art, with the single goal of optimizing the various engineering and security trade-offs for this one application. We believe in building the project upon a sturdy foundation of proven, peer-reviewed technologies wherever possible, although in some cases, we had to invent new technology that did not exist previously. With the exception of these innovative features, the primary contribution of Pastel is in how it combines these existing technologies in innovative ways, while carefully designing the economic incentives of the network so that the optimal behavior of network participants is to act honestly and in a way that is productive for the performance and security of the network.

2.3 Data Storage in the Blockchain

The introduction of Bitcoin in 2009 showed the world that it is possible to create a secure, distributed ledger through the use of proof of work and Merkle trees. Soon after the original Bitcoin software was developed, developers realized that it was possible to embed arbitrary data into the blockchain in a variety of ways. This was quickly used to upload various files to the Bitcoin blockchain, from image files of cartoon characters to the malicious embedding of computer virus signatures (designed to trigger anti-virus software into thinking the blockchain data files were infected). While there are many ways one can go about writing data to the Bitcoin blockchain, the basic principle is to send a tiny amount of Bitcoin to a large number of addresses. The trick is that these addresses are constructed in a special way, so that while they appear to be valid, regular addresses, in reality each address represents a small

chunk of the data that the user wants to write to the blockchain (because the addresses are selected in this way, the sender cannot know the corresponding private keys to the receiving addresses, causing such coins to be permanently lost or "burned").

Later, by referencing the corresponding transaction ID, another user of the network can retrieve the list of receiving addresses and thereby reconstruct the original file. This is a very powerful idea, since Bitcoin has proven to be a highly secure, immutable ledger. The potential rewards for anyone who could break the security of the Bitcoin system are so large that they have effectively acted as the greatest "bug bounty" in software history. The ability to write arbitrary data to the blockchain in this way means that we can ignore the details of how Bitcoin is implemented (which is highly complex and non-trivial, representing the work of hundreds of talented and expert software developers around the world) and instead work at a higher level of abstraction, where we can treat it as a cloud storage service like Dropbox, but one in which files can only be added– not deleted or modified. Any file written in this manner automatically inherits all of the security properties of a Bitcoin transaction: it cannot be forged or counterfeited, and it is truly immutable and irreversible.

2.4 Limitations of This Approach

The problem with this method of storing data in the blockchain is that Bitcoin was not designed for it, and it does not scale. Writing even a small file (say, under 100 kilobytes, or one tenth of a megabyte) in this way requires hundreds of separate transfers to the "fake" blockchain addresses. If each block in the system is only a few megabytes, it is obviously not remotely scalable to have thousands of users writing files of any significant size. Indeed, this scaling issue caused the Bitcoin developer community in 2010-2011 to become increasingly hostile to this non-standard use of the network, which many developers viewed as abusive and not in the "spirit" of the Bitcoin project, which was conceived of as a digital currency or "e-Gold". As a result, this approach is only practical for writing very small files to the blockchain.

Still, as we will see in this Whitepaper, it *is* possible to build a secure system despite these limitations. We can think of these small files as representing *tickets* of various kinds. These tickets describe the different functionality we require for the network: for example, art registration tickets, identity establishing tickets, artwork trade tickets, and so on. The tickets are written in a simple, JSON-like binary representation called *MessagePack*, and are designed to be compact and efficient. The two key ideas that make this possible are 1) file hash functions such as SHA3-256, which allow us to uniquely reference a

specific file without storing the entire file, and 2) a digital signature scheme, which allows us to cryptographically sign a ticket using a secret key so that any user can verify the authenticity of a ticket by checking the public key of the ticket author and verifying the signature against the public key. In this way, a piece of registered artwork cannot be "faked," just as one can't produce a counterfeit Bitcoin.

2.5 Off-Chain Storage

A system like the one described above, while secure, is not complete as an integrated digital art registry. That's because it leaves out the storage of the artwork itself, which is obviously a critical aspect of visual art. What is needed is an additional *off-chain storage system*, in which the original, high-resolution image files can be safely stored in a fault-tolerant, redundant, and decentralized way, so that they are always available to authorized owners– even in 10 years from now, and all without a centralized party or service underpinning the system. Pastel addresses this challenge through the use of a novel distributed file-storage layer, which we describe in further detail below.

We believe that if rare art on the blockchain is to have any hope of lasting relevance to the art market, it must address the issue of permanence. After all, we still have access to ancient Greek sculptures 2,000 years later, and even less permanent forms of art such as the drawings of Leonardo have reliably withstood the test of time. Yet anyone who has used computers over the last decades has a horror story about a dead hard drive that robbed them of their precious family photos or college term papers. Indeed, the only reason why we seem to have progressed past this stage of digital impermanence is the rise of cloud storage services, which are of little use in a truly decentralized system.

Indeed, we need to solve two challenges when it comes to off-chain file storage: the first is the *technical challenge* of how to make a reliable and secure system out of hundreds of unreliable nodes. The second is the *economic challenge* of how to price the various services required in order to compensate the network participants who store and process these files– all in a way that can adapt to changes in network conditions and to changes in the market value of the underlying coin. In this area, Pastel makes an original contribution in the form of a difficulty linked fee deflator factor, which allows fees to float up or down in response to market signals without introducing any external dependency to the outside world (e.g., a price feed from *Google* or *coinmarketcap.com*). We also emphasize the importance of a "pay once, store the art forever" pricing model in Pastel, unlike other decentralized storage projects such as *SiaCoin*, which requires periodic "rent" payments for the continued

storage of data in the system, which we believe stifles curious potential users by introducing excessive uncertainty as to the ultimate cost of data storage.

2.6 Inappropriate Content

Of course, storing the artwork image files "off-chain" is just a starting point; there are several other features that we believe are critical to a successful decentralized art registry. The first of these is a way to reliably prevent offensive or illegal content from being registered on the network in an "active" way, rather than responding passively through moderation when such content is identified by network participants. While this may seem like a questionable or "prude" feature, it is in reality quite serious: other decentralized networks, such as FreeNet, are notoriously plagued by child pornography and other illicit content. Not only are such uses of the network morally repugnant, they also introduce real legal risk to the owner of the computer on which the content is stored.

While determining if a candidate image is pornographic or otherwise "NSFW" is very challenging, recent advances in machine learning have led to open-source models that have been trained on millions of sample images. Pastel uses a pre-trained *TensorFlow* model from Yahoo's *OpenNSFW* project to evaluate every submitted art image file; the model assigns a score between 0 (definitely not NSFW) and 1 (definitely NSFW). By insisting on a relatively low score, we can ensure that the majority of inappropriate content will never be permitted on the network. While this is in effect a form of censorship, and will likely lead to the rejection of artworks that would otherwise be considered by a human observer to constitute "acceptable art" (such as a tastefully done nude scene), we believe that this sacrifice is well worth it to establish a baseline level of confidence in the nature (and legality) of the stored image content.

2.7 Duplicate Detection

Perhaps the most technically challenging feature required in any decentralized art registry is some form of *near-duplicate image detection*. The idea here is that we only want original works to be registered on the network. That is, if a particular image is registered on the network, then we don't want to allow another user (or even the same artist) to register a "near-duplicate" image. If we were only concerned about an exact bit-for-bit duplicate of the original image file, we could simply use a file hash, and insist that this be unique. But a file hash is brittle: if you take an existing registered image and change only the upper left pixel, for example, the entire hash will change. What we want is

the exact opposite: a *robust* image fingerprint– one that is stable in the face of superficial changes. That is, we want a way of identifying or characterizing an image that is robust to various transformations to the original image, which might include:

- Cropping, scaling, or rotating the image.
- Adjusting the color, contrast, brightness, or "curves" of the image.
- Adding random noise or dots to the image.
- Applying any sort of image filter, such as those included in Adobe's Photoshop software package; for example: blur/sharpen, edge-detection, inverted images, non-linear image warping filters such as Spherize or Twist, etc.

Put differently, we want a duplicate detection system that can react similarly to the way a human observer could in determining if two images are "related"; that is, if an average person could reliably determine that a given image is "excessively derivative" of an existing registered image, then we want our automated system to reach the same conclusion. The ideal system would reject a high percentage of true duplicate works while allowing through the vast bulk of truly original works. The greatest challenge are those artworks on the boundary line– similar to an existing artwork, but different enough that they are not clearly "duplicates" according to our chosen criteria. For this problem, Pastel has developed a novel and innovative duplicate detection framework, which leverages advances in machine learning technology as well as the creative application of classical statistical techniques, which we explain in more detail in a later section.

3 Market Opportunity & Economic Rationale

In this section, we discuss the market opportunity for the Pastel project in the context of the global art market. We will analyze the inefficiencies in the current market processes, and identify areas in which the introduction of decentralization would result in an increase in market efficiency. In particular, we will focus on how the advent of digital technology–especially the ability to make an exact "bit-for-bit" copy of a digital file–has changed the underlying assumptions that have historically guided art market participants.

The inability of the art market to adequately respond to these new challenges has created the current absurd situation for the sale and distribution of

"high end" digital artworks, in which artists sign legal contracts with buyers, ensuring buyers that the artist will only keep "one copy" of the digital file for the artist's personal files, so that the buyer can benefit from the exclusivity and rarity of the artwork. In many cases, such digital artworks are sold as complete, self-contained physical systems, including televisions and DVD players. But what would prevent someone (say, an unscrupulous maintenance worker at a gallery) from secretly copying the DVD? And if that were to happen, what might that imply for the value of the "artwork" that the buyer spent real money on?

3.1 Physical vs. Digital Art

In the case of traditional art objects, such as paintings, drawings, sculptures, and limited-edition prints, a "copy" is obviously inferior to the original object. As long as it is possible to distinguish authentic works from counterfeits, no rational buyer would pay nearly as much for a copy. But how should one think about a "natively" digital artwork– an artwork which the creator intended from inception to be presented in a digital format (as opposed to, say, a digital photo of a physical painting)– which is, in a literal sense, a specific series of zeros and ones. Is not every identical list of digits "the same" in a similar way to how all gold atoms are the same? Put differently, the physical instantiation of a natively digital work of art seems to be fundamentally secondary to its essence; who cares if it is stored on a DVD or a USB flash memory drive? Won't art historians of the future care more about the "data" itself, which is simply information without physical form?

An analogy is useful in understanding the differing dynamics of natively digital artwork. Imagine two scenarios: in the first, a person possesses of a mint condition original copy of the very first *Spider-Man* comic book. In the second, a person possesses a high-resolution PDF file of that same comic book. While both of these people would be able to read and enjoy the artwork featured in the object, one of them is worth thousands of dollars in the marketplace, while the other would be considered worthless by most. The reason, of course, is that the physical edition is *rare*. When Spider-Man was first invented, no one knew that it would later take on iconic cultural status: few were made, and of those that were made, most were discarded or eventually lost or destroyed by the children and adults who bought them.

Now, decades later, there are millions of Spider-Man fans around the world, and of these fans, thousands of them are likely to be extremely wealthy. And yet, there are likely only hundreds of mint-condition, certified authentic copies of Spider-Man number one in existence. Thus, through the fundamental eco-

nomic forces of supply and demand, the price of an authentic Spider-Man number one (as of 2018) is over $700k (Source: gocollect.com). But the PDF version of this comic-book– no matter how high-resolution the scan might be– is very nearly worthless. The reason for this disparity is that the person who possesses the PDF file can, for example, email it to a friend for free, and now the friend has the identical file. The ability of digital files to be endlessly duplicated, with no loss of quality, fundamentally undermines the supply side of the equation, and thus permanently impairs the value of the digital asset.

The "de-facto exclusivity" that goes along with possession of a physical object (i.e., it is in your house, so it can't also be in anyone else's house at the same time) is thus a critical factor in the value of traditional artworks. While this attribute comes "automatically" in physical artwork by virtue of the intrinsic qualities of space and matter, it is completely lacking in the digital realm. The purpose of the Pastel project, at its core, is to synthetically augment digital art files to bestow them with these same properties– a way to say, in an internally consistent, mathematically precise way– that one party (and only one party) owns a given digital artwork, or to memorialize when that party transfers this artwork to another party in a blockchain-settled financial transaction.

3.2 The Art Market

The global art market was estimated in a 2017 *New York Times* article to exceed $50 billion dollars per year. While much of that value is dominated by a few high-priced sales at auction of famous historical works that can exceed hundreds of millions of dollars, the size of the contemporary art market, as measured by using global auction revenue, is close to $2 billion annually (Source: artprice.com). In addition, the contemporary art market employs many thousands of individuals around the world, from the artists themselves to their assistants, art dealers, gallerists, art advisors (who advise collectors), museum curators, publicists, etc.

3.3 High-End Contemporary Art

The "high end" of the contemporary art market is increasingly dominated, at least in dollar terms, by a small group of "star" artists, such Damien Hirst, Jeff Koons, and Matthew Barney, who often employ whole teams of assistants to jointly produce their artworks. These famous studios work closely with the most reputable galleries and collectors, and cultivate social relationships with

important art critics and taste-makers. Without such connections, it is extremely difficult for young artists trying to establish themselves to get enough career traction so that they are able to financially support themselves through their art. Thus we find ourselves in a world where being a "working artist" increasingly means to be part of an exclusive social club, which prevents otherwise worthy artists from contributing for lack of such connections. Indeed, a young artist who does not physically reside in a small number of expensive cities, such as New York and Paris, faces a steep, uphill battle to even have a chance at establishing such connections. And for those artists who do manage to catch the attention of a gallery or dealer who is established enough to financially support the artist, the price is very high: dealers generally take the majority of the economics, and the additional layers of fees from galleries and art-advisors usually means that the artist is left with a small fraction of the total amount spent by the end buyer.

Why has the market evolved in this way? Much of it arises from the physical nature of art, and the fact that wealthy art buyers want to view their expensive potential acquisitions in a pleasant physical space– usually one with high ceilings and located in the trendiest sections of the most expensive cities in the world. Obviously, a typical "starving artist" will lack the resources to offer such a physical venue; furthermore, most artists don't begin their career with social connections to the wealthy people who mostly constitute the art buying market, and thus depend completely on the dealer or gallery to facilitate sales of the artwork and to promote interest in the artist's works– say, by holding exclusive gallery opening events with fancy wine and "VIP guest lists."

So it's not like the galleries aren't justifying their share of the economics: indeed, they are so important that artists are generally completely reliant upon them– which is why the artists are willing to give up so much of the theoretical revenues from selling their works direct to the end buyer (that is, the collector who ends up paying the highest price for the artwork). But this importance mostly stems from factors that are extrinsic to the art itself, such as the financial and social resources required to successfully market and sell the artwork. In the case of digital artwork, the physical aspects (such as having access to a fancy gallery space), are perhaps less relevant. And the combination of digital art with the Internet means that there is a much larger pool of potential buyers for that artwork.

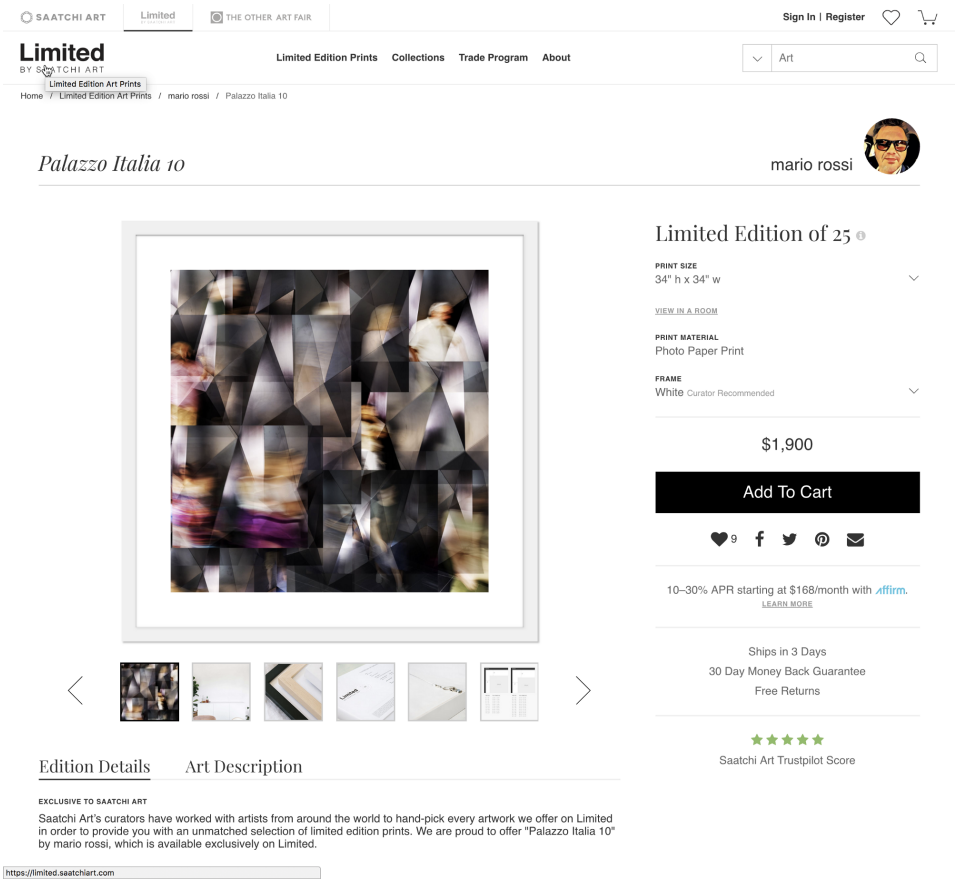
3.4 Eliminating Intermediaries

It's not just the art galleries that take a piece of the economics in art market transactions: there is a whole industry of art appraisers, who certify the au-

thenticity and attempt to estimate the value of specific artworks using various methodologies, similar to how a real estate appraiser might value a building. These art experts are generally hired by the owners of the art, making their conclusions potentially biased (e.g., an owner can "shop" for a favorably disposed expert, knowing that the negative views of the skeptical experts will likely never be seen by potential buyers) and unreliable. In particular, fraud is rampant across the art market, ranging from physical counterfeiting of known works to the creation from scratch of "fake" works that are nonetheless attributed by experts to an important dead artist. Any reader doubting this is encouraged to watch the 2014 documentary *Beltracchi* about a notorious art forger who is alleged to have fraudulently sold over $50 million worth of "fake" artworks that he skillfully made from scratch and for which he created elaborate and believable phony back-stories, or to read about other historical forgers, such as Han van Meegeren or Elmyr de Hory. Limited edition prints, which, given their method of creation makes them inherently more vulnerable to forgery, are particularly risky from the standpoint of inexperience art collectors.

Yet, the art industry continues to extract a huge portion of the total value chain in the art market, even for digital artwork. The figure below shows the website of a famous and respected contemporary art gallery, *Saatchi Art*:

Figure 1: An Example of the Contemporary Art market (Source: *www.saatchiart.com*



The example in the figure suggests that at least some buyers are interested in purchasing one of 25 limited edition prints for nearly $2,000. What does the buyer get for all that money? A key part of it is the confidence that, as a buyer from a reputable dealer such as the *Saatchi Gallery*, you will be protected against outright fraud and deception. That is, that the dealer is vouching for the claim that there truly are only 25 such prints in existence, and has done the requisite work to determine that the print offered for sale is an authentic member of that limited collection and thus actually made by the artist in question. That's why any purchase from a reputable gallery comes with a set of supporting documents whose purpose is to conclusively establish the authenticity and provenance of the work– something that is particularly important if the buyer ever hopes to resell the artwork in the future to another collector.

3.5 Intermediaries Take from Both Sides

Even in the case where the buyer has "paid up" for the artwork in order to get a higher level of assurance and confidence that the artwork is legitimate, what happens when this buyer later wants to sell the artwork–say, because of what are known in the art industry as the "three D's": debt, death, and divorce? The normal avenues for selling second-hand goods, such as *eBay*, simply do not work for rare art; even with a folder of supporting documentation received from the gallery, what rational person or company is going to trust some random art collector that the work is genuine? It's not like it's so difficult to fake such documentation, especially if the fraudster has access to the actual documentation from a legitimate copy of the artwork. After all, the next buyer isn't going to get the benefit of purchasing from a reputable dealer, since in this example they are buying it directly from the original buyer.

In reality, the buyer who purchases a piece of artwork from a gallery is usually helpless to efficiently sell the artwork without the cooperation of a gallery (often the very same gallery that sold it in the first place), which again comes at a very steep cost in transactions fees and the co-called "bid ask spread," or the difference between buying and selling prices that exist even when artwork is sold from one dealer to another dealer. These frictional costs reduce the incentive to transact in the art market, and the resulting loss of liquidity hurts both artists and collectors.

If we return to the analogy of comic books, this dilemma would be familiar to anyone who has ever purchased a "rare" comic book from a comic book store (say, a $50 issue) and then, years later, attempted to sell the comic book back to the store (or to another comic book store): the hapless buyer would be lucky to get even $10 from the store for the supposedly rare comic book. The same can be said about most collectibles, be they *Magic: The Gathering* cards or *Beanie Babies*. The relative disadvantage of one party versus the other (i.e., the comic store can sell to many local customers, but the typical customer can only sell to a few local comic stores) explains much of this market inefficiency. By inefficiency, we mean that market participants (buyers and sellers) as a whole could be better off– if only they could identify and transact with each other. But because this is prevented by various reasons, collectors are forced to transact through intermediaries which end up extracting most of the "economic rent".

In short, the art market, as well as the rare collectibles market (e.g., comic books, baseball cards, etc.), is rife with inefficiencies for both consumers and producers. The people who are actually making the art are getting a fraction of the final value created, and the people who are actually buying the art are

also forced to give up an enormous share of the potential profit through the various layers of fees and the lack of liquidity. The reason why collectors put up with this untenable situation is because the art market has *always* worked in this way, and until now, there have been no good alternatives that retain the "reputation backed security" offered by the traditional gallery or retail model.

3.6 Digital Art Can Be Different

While the inefficiencies that exist in the market for physical artwork are not going to change anytime soon, there is no reason why the new world of digital artwork has to be shackled to this inefficient and inequitable system. But before that can happen, the world needs a decentralized, *trustless* mechanism to fulfill these same core functions of the art market that are currently provided by galleries and experts. Why must it be decentralized? In short, no one is going to trust a centralized network with something as important as high-end (or even low-end) artwork or valuable rare collectibles. What artist in their right mind would want their art to be held hostage in some "closed-garden" environment controlled by a corporation or government?

And what rational buyer would want their purchases to be subject to the whims of the operator, who would be free to dictate which works were authentic and who owned which works? What would prevent such an all-powerful art industry participant from abusing that power to extract value from artists and collectors? The art world is already dominated to a large degree by the major auction house companies (e.g., *Sotheby's*, *Christie's*, etc.), which have repeatedly been found guilty of engaging in price-fixing schemes and other monopolistic practices, and the contemporary art market is similarly dominated by a handful of powerful dealers who often have exclusive relationships with important artists. The last thing the art world wants or needs is to cede even more power and control to a single entity. And collectors simply do not want to own their collections "at the sufferance" of an all-powerful organization, which could theoretically "take back" artwork that was purchased in the same way that *Apple* could remove a downloaded movie file from a user's *iTunes* account.

Even if we could be blessed with a hypothetically "benevolent" centralized operator that never abuses its position of power, huge risks would remain. What if the operator is unable to maintain the centralized system, say because of bankruptcy, or a change in laws? What would happen to all of the artwork that had been registered in such a centralized system in this case? The potential situation here is analogous to the cautionary tale of *GeoCities*,

a pioneering website in the early days of the Internet that allowed users could create free personal websites. The company running the service abruptly shut down the website, causing the permanent loss of a large amount of potentially culturally significant content. Because the service was centrally controlled, there were limited options for those in the community who wanted to preserve these files. Aside from attempting to download the data while it was still available and creating a mirror for it, outsiders were at the whim of the decisions of company insiders.

Contrast this with a decentralized project such as *Wikipedia*; we never need to worry that the organization behind the operation of Wikipedia will give up and shut down the site, since others could step in to provide the required funding and operational expertise, given that the entire system is open-source. At the same time, there is some degree of unavoidable centralization involved in this scenario. For example, the web domain, *wikipedia.org*, is not freely available for anyone to direct to a particular IP address– it is privately controlled by the board of trustees of a non-profit foundation. Still, the open-source nature of Wikipedia means that an unaffiliated third party could create a clone of the site and host it at a new URL. If the new version worked as well as the old one, and the old one were hypothetically no longer available (not a realistic scenario, since Wikipedia is thriving), then it is reasonable to suppose that users would switch over– thus preserving the essential functionality of the service indefinitely. That is, users are free to help themselves if they so require, and they don't need to ask for anyone's permission to do so.

In a fully decentralized peer-to-peer system such as Pastel, where all of the software is open-source and anyone can freely purchase the coins required to "host" the network (i.e., to run a Masternode) so that it can serve users, the community is never faced with this problem. If the original Pastel developers were to abandon the project for some reason, anyone (say, an artist or art collector) could carry on the operation of the system. The essence of a decentralized system is that no one is in control, which gives individual network participants a degree of freedom and power that is simply unavailable in a centralized system; centralization means that users must act passively, with no control over their own destiny. This is what gives us at least some chance to fulfill the implied guaranty of our proposed system: that it will securely store users' artworks "forever". After all, any owner of a large number of artworks registered on the network would have a personal motive to continue running the Masternode software, even if doing so would otherwise be financially undesirable. And a new group of leaders from the outside could become "activist" and publicly announce that they are going to "take over" the system; if these outsiders can make a compelling enough case that a majority of Masternode

operators are convinced to give them a chance, no one can stop them– even the original Pastel development team.

3.7 An Alternative View of the Opportunity

Thus far in this section, we have focused on the glamorous world of "high end" artwork, which is an unfamiliar setting to the vast majority of people alive in the world today (despite the fact that it represents the majority of the dollar value of art transacted each year). There is a much larger world, in terms of the number of artists and potential collectors, if one looks at lower priced transactions, which often take place in niche communities. For example, there are a huge number of fans of Japanese style *anime* artwork, and well-known anime artists can have tens of thousands of followers on social media– fans who are often dedicated to a particular artist, following the artist's personal story in addition to the artwork itself.

Yet, even a well known anime artist, to continue with that example, faces a difficult task in converting such fan interest and activity into meaningful economic value. While the artist can create single drawings of characters and sell these to fans over the Internet, given the dynamics and attributes of the fan base (i.e., a large number of fans that are unlikely or unwilling to spend a large amount of money on a drawing), this is not likely to result in a significant amount of revenue. After all, most of these artists are popular on social media *because* they regularly post their works free-of-charge on these forums. If the fans are already getting the artworks for free, why would they want to pay for them? Thus, such an anime artist must usually find other means of financially supporting themselves. For example, the artist might do contract work to produce artwork for a video game or movie project– but then the producer of the content tends to take most of the ultimate value paid by consumers of the content, leaving little for the artist.

There are, of course, other ways to monetize artistic talent today. Savvy artists who are able to cultivate a large community of fans on the Internet can earn money through advertising or affiliate marketing arrangements, or by receiving commissions for custom work from fans (although the latter is not particularly scalable since custom works can generally be sold only a single time). The introduction of such Internet services as *Patreon*, which allows fans of an artist (or more generally, any content creator) to become a virtual "patron" to the artist, usually accompanied by some form of acknowledgement of the support of the patron by the artist– say, by listing the patron's name at the end of a video. While this form of "crowd-funding" artwork is rapidly growing, creating a new economic model for successful content creators, it is

severely limited due to the so-called "free rider" problem: essentially, these services allow creators to act analogously to a musician busking on the street– anyone can pass by and listen for free. Unless a critically large group of fans can somehow be moved to provide compensation, the activity is unlikely to be particularly profitable for the artist or even be financially sustainable without an additional source of employment for the artist.

So why do fans nevertheless give money to creators through services such as Patreon? Although in some cases, these fans may believe that their contribution is the deciding factor as to whether the content will be created or not– making the contribution akin to a purchase– most do it more out of an appreciation for the content and a desire to support or assist the creator. That is to say, the "buyer" in this situation generally doesn't receive much tangible value in return– it is more like a form of charity, with the primary benefit being psychic in nature: the fan can feel good about helping the artist. While this is no doubt a wonderful phenomenon, it cannot be expected to generate much in the way of money except for the most popular artists, since a very small percentage of fans will in general be willing to give what is essentially a donation, and those who do will tend to give only a small amount of money.

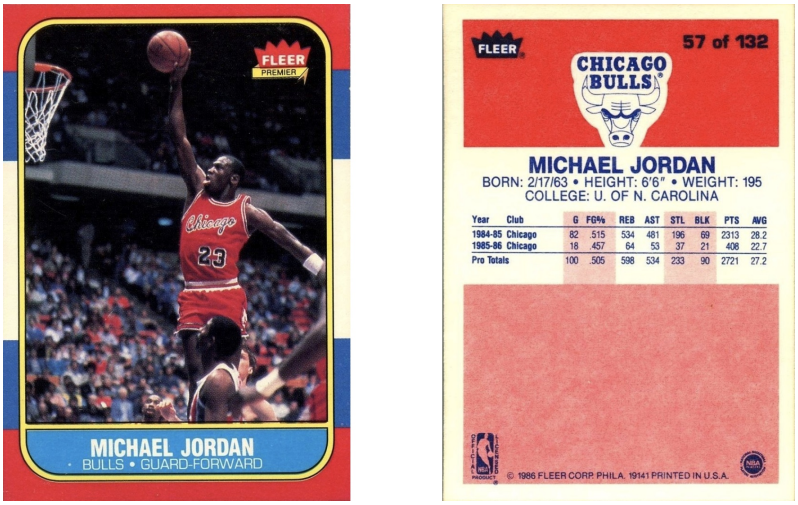
3.8 Crypto-Currency as a Solution

What if there were some way to give the fans of an artist a way to support the artist financially, while still deriving some tangible value themselves from the transaction? That is, if instead of simply giving a donation, the fan could receive something in return for their support? Of course, there are some obvious "low tech" ways to accomplish this today; for example, the artist could simply physically create "limited edition prints" and offer to ship these to contributors in return for a contribution. But this doesn't work well in practice; there is a lot of overhead created, with the cost of shipping and printing the artwork setting a relatively high minimum price level below which the transaction doesn't make financial sense for the buyer or seller. Worst of all, the collector who receives such a physical print in return for their contribution will face an almost complete lack of liquidity. That is, if collector ever wants to sell it, they will be lucky to get even a few dollars on *eBay* or other marketplaces, since no one else would be able to readily assess the authenticity of the work, and even if they could, the probability that a seller would be able to reach the potentially tiny audience of potential buyers through an unpublicized listing on an auction website is remote at best.

We believe that rare digital artwork, implemented correctly on a decentralized blockchain, offers a better solution. By giving the collector the ability to

transfer value directly to the artist with minimal fees–that is, by the allowing the collector to purchase one of the rare "copies" of the artwork offered for sale by the artist, using an integrated, decentralized asset exchange– that we can offer a far better way to create a mutually beneficial exchange between artist and collector. Savvy collectors, who have a good eye for talent and their "finger on the pulse" of popular tastes, could find an artist just starting out with few fans, and inexpensively purchase one of say, 10 limited editions of one of the artist's digital artworks. If that artist were to later become a huge celebrity, with thousands of fans (after all, some of the biggest celebrities today got their start from humble origins on social media), then it's easy to see how such early artworks could appreciate in price dramatically– say, from $4 to $100.

Figure 2: Michael Jordan Rookie Card (Source: *the Fleer Corporation*)



The scenario outlined here is exactly analogous to a knowledgeable basketball fan deciding in 1986 to purchase a box of baseball cards in order to acquire 10 copies of Michael Jordan's rookie card, shown in the figure above. Such cards now fetch as much as $2,000 in the market, although cautionary tales of fake counterfeit cards abound given the ease of reproducing a near perfect clone of a paper card. Although there is generally no reward for being an early fan of an artist who goes on to later mainstream success (other than the satisfaction of seeing one's tastes vindicated), in our proposed system, a fan can actually benefit financially, thus creating a symbiotic relationship between the fan and the artist:

- The fan either gives coins to the artist in return for the artwork (which

the artist can sell for Bitcoin on an exchange) or purchases the artwork in a secondary market transaction from another user (i.e., if there is another user offering it for sale at a cheaper price than the artist is offering, available via the integrated decentralized exchange; since every copy is exactly the same, users would likely prefer the cheapest unless they are specifically trying to support the artist with their purchase). Thus each purchase financially benefits the artist, either by directly transferring value to the artist or indirectly by increasing the market demand and price for the artist's works on the network.

- The fan now owns a liquid asset, denominated in a crypto-currency which can be converted into Bitcoin (and thus ultimately into any fiat currency), the value of which depends of the success of the artist. Thus the fan or collector now has a vested interest in promoting the works of the artist, since the artist's future success directly benefits the collector in the form of a higher market price for the artist's works. Ultimately, the early "backers" of a talented but unknown artist could see their contribution repaid many times over through the market price appreciation of their rare editions of that artist's digital artworks– artwork that the collector will be able to sell directly to other users with minimal brokerage fees and without the risk of payment fraud, since the integrated art trading system uses a trustless escrow system.

The key advantages of rare digital art, and why it is such a potentially transformative force in the way art is collected and created, include the following:

1. Since the art exists in digital form (i.e., as information), **the costs to create it and distribute it are much lower than with physical artwork**. We can create 10 or 100 copies of an artwork just as easily, and each one is exactly equivalent to all the others. We don't have to deal with fancy real estate (we still could, of course– nothing prevents our proposed system from working in conjunction with a gallery, which could handle the registration and trading aspects on behalf of artists or collectors) or high security, climate-controlled storage facilities to safeguard the art from theft. And we don't need to deal with expensive specialized moving companies, or parcel services such as UPS in the case of "lower-end" artwork; instead, we can simply send the art to another person just as one might send a Bitcoin– and after it has been sent, this "electronic shipment" is just as substantive and irreversible as if we had in fact sent a physical object: we would no longer have the artwork, and

the recipient would. And we also wouldn't need to worry about the art deteriorating in some way over time, since the digital content is abstract. Since ownership of a digital artwork, like the ownership of Bitcoin, ultimately amounts to knowledge of a certain private key, the owner of a digital art portfolio can move to a new country without ever dealing with the complex process of transporting the work and dealing with customs agents.

2. The use of secure cryptographic methods allows rare digital art to **solve the problem of establishing the authorship and provenance of a given artwork**, which has plagued the art market for centuries. Our proposed system, if used correctly by both artist and collector, prevents counterfeiting and impersonation– issues that are partly responsible for the high levels of fees and intermediation found in the traditional art market. Thus, some of the time, energy, and money that is currently spent to protect art buyers from fraud can instead go towards compensating the artists, and also, indirectly, the owners of the digital artwork, in the form of higher purchase prices. No longer do hapless buyers need to discover years after their purchase (when they try to sell their collected art) that it was never genuine in the first place. And artists wouldn't need to worry about impostors going around claiming to be them, potentially damaging their reputations as a result of inferior quality work: as long as the artist has kept their Pastel ID private key secure, no one else can generate the digital signatures required to make an unauthorized image appear to be created by the artist, since this is protected by provably secure cryptographic signatures. Buyers are similarly protected, in that they can seek out additional verification (e.g., by checking the artist's website or *Instagram* account to verify the artist's public key, or to ensure that the artist hadn't disclosed a security problem with their private key, which might be accompanied by a warning from the artist to fans/collectors that they should no longer trust new works issues by that ID) and use this to determine with absolute certainty that the artwork they are contemplating buying is genuine.

3. By integrating the payment and trading aspects of art collecting into the very same blockchain that acts as the registry for that artwork, we greatly **reduce the need for extensive intermediation in the form of galleries, dealers, and payment processors.** This gives more of the economic value of the art market to the artists and the collectors instead of to middle-men, leading to more vibrant art market

and hopefully resulting in more art being created and consumed. Also, by reducing the frictional costs (in terms of both time and money) of transacting in the art market, we lower the bar for new art collectors to enter the market. There is no doubt still room for some intermediaries in the art market, such as critics and advisors that can help collectors discover artists that the collector might enjoy, but in a system where the end buyer or seller can directly access information and transact securely, such intermediaries will need to continually prove their value in order to retain their clients– it would be much harder for intermediaries to "coast" on their reputation and positive, extracting value merely by exploiting art market inefficiencies.

4. By giving a new way for "casual fans"– that is, non-wealthy consumers who enjoy the works of an artist– **to financially interact with their favorite artists through efficient micro-payments, rare digital art enables new ways for artists to support themselves** instead of using services such as *Patreon* or *GoFundMe*, or by taking on contract work for hire that they do not own. Particularly for users living in low cost geographies, where wages are extremely low by the standards of the developed world (e.g., the Philippines), the ability for a talented artist to sell 50 copies of an rare digital artwork for, say, $2 a piece, is potentially life altering: it could mean the difference between being able to pursue one's dream or being forced to take other employment to make ends meet. In a broader sense, this innovation can have an "ennobling" impact on the global art market, by allowing worthy individuals to participate on a global stage. Imagine a world in which an unknown poor resident of a third-world country could be potentially found next to the work of a trendy Western artist, with both hosted on the same system. Perhaps the end result would be a more egalitarian and less elitist art world.

5. By giving collectors the opportunity to share in the financial success of their favorite artists, we create a new set of positively self-reinforcing incentives, similar to how a sports fan's level of engagement towards a particular game is higher if that fan is gambling on the outcome of the game. But unlike in a sports game, where the fan must sit idly by even as his team is losing, **the fan in the rare digital art context can impact the probability of success by promoting the artwork in various ways** (say, by writing a thoughtful review of the artwork in a respected publication, or by hosting a physical showing of the work at an event). Collectors would also have an increased incentive to seek out

new works by up-and-coming artists, dedicating more time and energy to browsing the latest submissions from around the world in the hopes of finding the "next big artist." This could make what might otherwise be "passive collecting" into something more akin to activist investing in the public stock market.

6. Because of the intrinsically public nature of a Bitcoin-like blockchain, digital art of the form we propose would **dramatically improve the transparency of the art market**, with buyers able to see the entire history of trades for a given work. The art market is famously opaque, with few participants truly understanding how much of each dollar is going to the different parties involved in a given transaction. The same physical art print might sell for wildly different prices in different countries at the same time, without either buyer ever finding out about the other transaction. Unlike the stock market, there is no global reporting of trades in the art market, which creates extreme information asymmetry– something that primarily benefits knowledgeable insiders– generally the very same middle-men who already extract much of the value. In our proposed system, every trade of a given artwork, including the quantity sold and the price of the sale, is recorded in the form of a trade ticket in the blockchain. This means that buyers and sellers of digital artwork can make informed decisions with access to hard facts– how much other buyers and sellers were willing to pay at various times– so they can reach a view of the overall trend of value over time for a given artwork or artist.

7. The same properties that make digital art easier from a transportation and logistics standpoint also mean that digital art is potentially much **more resistant to censorship than physical art**. Just as a powerful government would be nearly helpless to reverse a valid Bitcoin transaction, a properly implemented digital art registry on the blockchain (together with a sufficiently secure and redundant off-chain storage system as detailed below) would make it extremely difficult for a hostile actor (say, a copyright holder, or a government censor) to forcefully remove a given artwork from the system or prevent users from viewing it or trading it with each other. Unlike stock certificates, which can be seized from a person's account by court order, the rights and control over one's rare digital artwork cannot be taken without the private key being stolen, much like the difference between a traditional bank account and a Bitcoin "paper wallet." While theft of private keys is certainly possible, users can at least measure this risk and proceed only if they are confident

that they can manage or tolerate these risks.

8. Not only does rare digital art make it much easier for a collector to protect against fraudulent or counterfeit works, but it also protects the collector from unscrupulous artists who betray their "promise of rareness." In the real world, an artist could easily decide to make additional copies of prints– or they could change a tiny detail and make a print of that instead. While such acts would probably hurt the artist's reputation, it can and does happen in practice. But the powerful duplicate detection system built into the Pastel architecture would protect the potential collector against an artist who attempts to register the "same" artwork; even if the artist were to modify the image in various ways, it would still be rejected by the network as a duplicate if it is excessively similar, which we will explain in more detail below. This means that collectors no longer have to rely on anyone for guaranteeing the continued rarity of their art collection– it is trustless, just as the ownership and and trading of the artwork is trustless.

9. The fact that digital art assets are built using cryptography endows them with special powers that wouldn't exist with traditional physical artwork– for example, the ability to indisputably prove to anyone that you own a given artwork without losing control of the art (this might be useful, for example, in purchasing property insurance on the artwork). This can be easily done through the use of digital signatures: the rightful owner of an artwork would simply generate a signature proving that they know the private key– all without ever revealing this key to anyone, and without leaving the comfort of their own home.

4 The Masternode Concept

We have now outlined the various pieces of functionality required for a successful digital art registry, but many questions remain. Who will provide this functionality to the network? And why should they contribute their computational resources to enable this functionality? And how can we protect the integrity of the network in the face of malicious network participants? We believe that all of these questions are answered by the concept of a *Masternode*, or MN for short.

4.1 History and Overview

The idea of a Masternode was first introduced by Dash in 2014, and it is a critically important component in the design of Pastel. Unlike in Bitcoin, where the "full nodes" that validate transactions do not participate in the mining rewards (i.e., the miners in Bitcoin receive all of the newly minted coins), Dash allows for certain nodes to take on a privileged position in the network, and in return receive a portion of the mining block rewards– but only if the operators of those nodes can prove that they control a sufficiently high number of coins. In the case of Dash, this is 1,000 DASH coins, which represents many tens of thousands of dollars. In addition to the requirement that a Masternode operator own a certain number of coins, MNs are expected to provide various additional services to the network, such as "instant" transactions in the case of Dash. Furthermore, the machines running the MN software must be hosted at static IP addresses and must constantly respond to pings and other "challenges" in order to prove to the other network participants that they should retain their MN status and share of newly minted coins.

In Dash, the Masternodes as a group receive 45% of the total block reward, with each valid MN receiving an equal share on average. As in Bitcoin mining, there is a zero-sum game aspect to being a Masternode: the more MNs that are active on the network, the lower the payment will be to each MN. Thus the *return on investment*, or ROI, of a Masternode– computed as the run-rate estimated number of coins produced in a year, divided by the number of coins required to establish an MN– will decline as the MN network grows (and conversely, the ROI will increase in percentage terms as the number of active MNs declines).Yet this decline in ROI as a result of a higher number of active Masternodes is at least partially offset by the market impact of users purchasing enough coins to qualify for a MN, which presumably would raise the market price of the underlying coin.

This points to a key benefit of the Masternode concept: it benefits both the ***economics*** of the network, by producing an economic return and incentive structure for the providers of services to the network, while also benefiting the ***security*** of the system by solving the problem of what is known as a *Sybil attack*, which essentially means that an attacker can create the illusion that their actions are the result of a large number of independent users, when in reality these "independent" users are simply guises of the single attacker.

A Sybil Attack can occur in any decentralized peer-to-peer networks when it is "cheap" for participants to create multiple identities, or "sock puppet" accounts on the network; the requirement that a MN operator own a large number of coins makes each identity expensive, which means that it would re-

quire a huge investment to run a large enough number of MNs on the network. Not only does this make an attack much harder to pull off, by eliminating the threat from unsophisticated "script kiddie" type attackers operating out of their parents' basement, it also wouldn't make sense for a rational attacker. After all, if the attacker controls enough MNs to mount such an attack, then by definition the attacker has a large financial stake in the network; thus, it would be self-defeating to attack the network. Even if the attack were successful– and it bears mentioning that we have designed various safeguards in to the software to mitigate such attacks if they were to occur– the resulting loss in confidence in the network and coin would cause the attacker to suffer a large financial loss which would probably exceed any ill-gotten gains made from the attack in the first place.

4.2 The "Crypto-Economics" of Tokens

The single most important benefit of a Masternode system is that it provides an answer to the question: *"Why would anyone want to hold this coin over the long term?"* The response to this question is ultimately what determines the market value of a crypto-currency network. In Bitcoin, this question is answered clearly, in that the rate of production of new Bitcoin declines exponentially over time, and thus if it proves to be a partial replacement for the gold market, one might rationally extrapolate that the value would be higher in the future as any growth in demand is offset by the decrease in available supply. After all, there will never be more than 21 million Bitcoins in existence, so that even if we focus only on the richest 1% of people globally (approximately 70 million people), there are not enough Bitcoins for them to all own a whole Bitcoin. But what about a network such as Pastel, which is attempting to facilitate other kinds of economic transaction among network participants besides simple transfers of value?

The problem with the majority of existing crypto token assets is that they utterly fail to answer this key question, and instead fall into what we can call the "Chuck-E-Cheese dilemma." This term is based on a comparison to *Chuck-E-Cheese*, a chain of arcade and pizza stores popular in North America that forced customers to pay for arcade games using the company's own proprietary tokens instead of regular (USD) coins, as illustrated in the following figure. If one considers the incentives of a parent hosting a child's birthday party, the parent wants to purchase only the minimum number of coins required to get through the party, and then wants to get rid of the remaining coins at the end.

That is to say, no one would rationally choose to hold on to these tokens over the long term, since there is absolutely no benefit to doing so– only down-

Figure 3: Chuck-E-Cheese Tokens (Source: *forums.collectors.com*)

side, in the form of reduced liquidity, since the tokens are useless anywhere besides that particular store, and if the company were to go bankrupt, the tokens would be as worthless as a *Circuit City* gift card (to name another defunct retail company). Any token that follows this basic structure– and unfortunately, this category describes the vast majority of "ERC-20" tokens that have been launched on the *Ethereum* platform– will have its fundamental value limited by the average value of transactions on the network at a given time.

In the case of the arcade company analogy, this value might be approximated by some discount to the equivalent dollar value, over a given time period, of all the games played in its stores if they instead required regular coins rather than proprietary tokens to play. That is, **the value of the coins are limited by the marginal utility value of the network**. This creates a "prisoner's dilemma" problem, in which users are motivated to hold the token for the shortest time possible, since others will come to the same conclusion and race to sell first, similar to how consumers tend to behave in hyper-inflationary economies such as Venezuela or Zimbabwe.

4.3 Masternodes as the Solution

The Masternode concept offers a potential response to this critical, all-important question: one might want to hold on to the coin rather than sell it quickly because if you can acquire enough coins to establish a Masternode, you can derive "dividends" or income in the form of your share of the newly minted coins generate in the network through the mining process. This arrangement can be viewed as a bond-like income stream, in that a certain number of newly

minted coins will come to the owner of the MN; if this number is sufficiently high, it could offset the impact of a price decline for the coin (e.g., if the Masternode ROI of the coin is 50% per year, and the coin declines in value by 30%, the MN owner is still in a profit position). And if the price of the underlying coin were to increase, then the financial returns would be even greater to the MN owner since they would have even more coins at the higher price.

The appeal of so-called "income investments" is well known in financial markets. It is a quirk of human psychology, and often results in irrational behavior, but investors enjoy "knowing that that they are only spending the income," rather than reducing their principle investment over time. Indeed, this propensity towards income often results in investors being sold a false bill-of-goods, where their "dividends" is actually just a return *of* their principle, rather than a return *on* their principle. Examples of this abound in markets, including dubious oil and gas "MLPs" and private Real Estate Investment Trusts that are really overpriced collections of low-quality real estate.

In addition to the income component of a Masternode, in the Pastel network we introduce further reasons for why users might want to hold the coin over the long term. Our essential strategy here is to make the operation of a Masternode– when conducted honestly and reliably, as regulated by a decentralized reputation tracking system– a gateway to a series of money-making opportunities that are directly based on the resources that the MN provides to the network. For example, Masternode operators are paid a registration fee to inspect and store art images, and are also paid brokerage fees for facilitating trades of digital artworks among users. These fees are in addition to the baseline economic return to the MN from its share of the mining block reward.

4.4 Masternodes are Decentralized

The architecture of Bitcoin can be viewed as one extreme in the spectrum of possible crypto-currency project designs: one in which each full node is essentially the same in terms of its power or influence over the network. The only way to get more influence in Bitcoin is by contributing more computational resources to the network. It is this architecture that gives Bitcoin its unparalleled degree of decentralization. In comparison, *Masternodes represent an intentional trade-off or compromise in which we accept incrementally higher levels of centralization* (since Masternodes will represent only a fraction of the total users of the network, but will wield a disproportionate level of influence on the network) in return for additional network functionality and improved economic dynamics (i.e, a higher incentive to hold the coin and operate the

network).

The reason why we should be willing to accept such a trade-off (and make no mistake, it *is* a security trade-off, in that any newly minted coins that are diverted by the network away from the miners that are actually contributing hash power to the proof-of-work of the network and sent instead to other network participants necessarily reduce the incentive for miners to invest their computational resources; this leads to a lower network hash-rate, which in turn makes the network more vulnerable to *51% attacks*, which can lead to *double-spending*– a potentially disastrous outcome for the value of the underlying coin) is because it facilitates greater network functionality by giving an incentive for network participants to provide such services. Because Bitcoin is so simple, being concerned only with the transfer of value between users, it does not require explicit compensation to convince users to operate full-nodes.

In contrast to Bitcoin, a system such as Pastel places much greater demands on the resources of MN machines: many of the services provided by Masternodes, such as duplicate image detection, require intensive computations. The storage of the art image files themselves in the off-chain storage system, once these files have been "inflated" so as to offer sufficient redundancy, also places significant demands on the disk space and bandwidth of Masternode machines. In particular, to make good on the promise to users that their registered artworks will be safely stored "in perpetuity," the network must explicitly compensate the providers of such services. In order to do this while still retaining the decentralization and security of the network is challenging, but the Masternode concept helps us here: it means that the network has a known group of machines, located at static IP addresses, that can be held accountable to the network. If they fail to provide these services, individual Masternodes can be financially penalized by the network by withholding some of their future share of the block reward. This functionality is built into the underlying Dash Masternode code used in the Pastel software.

The reason why a Masternode system can still be considered decentralized really boils down to one word: ***permissionless***. That is, anyone who wants to can set up a Masternode and help run the network– they don't need to ask for anyone's permission or approval, they just need to be able to come up with the money to purchase enough coins and to have the basic technical knowledge required to set up the machine, which is trivially easy to do now with freely available step-by-step guides that even a novice user could follow. As long as the coins can be freely acquired by market participants, either through purchase from an exchange for Bitcoin, or earned through the proof-of-work mining, or earned through the sale of rare artwork, then no one can prevent such participants from acquiring the necessary number of coins to earn

a "seat at the table" in the operations of the network by running their own Masternode. Even if the ownership of Masternodes is relatively concentrated at the inception of the network, over time, as the initial large investors seek to realize gains by selling their coins, this concentration risk naturally reduces over time as new participants enter the network and the number of independent Masternode operators increases.

4.5 Implementation of Masternodes

The basic Masternode architecture used in Pastel is heavily based on the Dash project, which is itself built on top of the Bitcoin architecture and code base. The way this is implemented is simple and elegant. To understand how it works, we first briefly review how blocks are produced in the Bitcoin network.

A new Bitcoin block is deemed valid by the network of full nodes if and only if the block is solely comprised of valid transactions (i.e., transactions where the sender is authorized to transfer the coins from the sending address, and the sending address contains a sufficiently high balance of coins to facilitate the transfer), and if the discoverer of the block (i.e., the miner) has managed to discover a "nonce" value that results in the block hashing to value with a sufficient number of leading zeros. This is simply how the proof-of-work mechanism functions in Bitcoin; since such nonces are very difficult to find, each miner is essentially buying a series of lottery tickets, and the more compute power each miner brings to bear, the larger the number of lottery tickets they are in effect buying.

The final requirement to mine a valid Bitcoin block is that the very first transaction listed in the candidate block is known as the *Coinbase* transaction (not to be confused with the centralized exchange company of the same name), and it is this special transaction that is responsible for the creation of new coins– the only way in which new coins are created in the network. That is, the Bitcoin network creates "out of thin air" some quantity of new coins, all of which which are then sent through the Coinbase transaction to the address selected by the miner who assembles the winning candidate block as compensation for performing the proof-of-work required to discover a new block (i.e., the mining block reward). That is, the miner receives 100% of the block reward, and this fact is memorialized as the first transaction in every block of transactions.

In comparison, the Dash system requires that a valid block meet an additional requirement in addition to those required for a Bitcoin block: similar to the Coinbase transaction, the second transaction in the block must send a por-

tion of the block reward to a designated Masternode recipient. Without this transaction, the block would be rejected as being invalid by network participants. But which Masternode should receive that payment? Unlike in the case of miners, where the average distribution of coins to miners is proportionate to the total computational resources demonstrated by the miner as a percentage of the total compute power of the network, the Masternodes are all "equal" in terms of their share of the network economics– at least in the absence of any potential penalties that might be assigned to Masternodes found to be guilty of bad behavior through the application of the reputation management system. Of course, one person or entity could simply buy enough coins to establish multiple MNs, which would give the owner a commensurately higher degree of influence on the network.

But as long as each MN owner controls a relatively small percentage of the total number of MNs, their ability to unduly influence the functionality or conclusions reached by the network overall is limited by the presence of the other MN owners (assuming these other owners are in fact independent). But the only way to get the huge number of coins required to potentially effect "the course of events" on the network is to purchase these in the open market– that is, through voluntary exchange. It's also important to note that even such a "whale" investor would still not have control over the underlying transactions of the network, since these are controlled only by the miners performing the proof-of-work for the network; this is the fundamental reason why such a hybrid network design is more secure and decentralized than a crypto-currency network that relies only on so-called *proof-of-stake*.

4.6 Rank Ordering of Masternodes

In order to make the Masternode concept work in practice, we require some mechanism for the Masternodes owners as a group to decide in a fair and secure way which Masternode should receive the next allotment of the block reward. More generally, we need a way to produce a rank ordering of all Masternodes for each new block added to the Blockchain. Whichever mechanism we choose for this purpose, it should satisfy two key criteria:

1. It should be *deterministic and thus able to be easily computed by any full node* on the network without making reference to the outside world, with each node independently reaching the exact same answer.

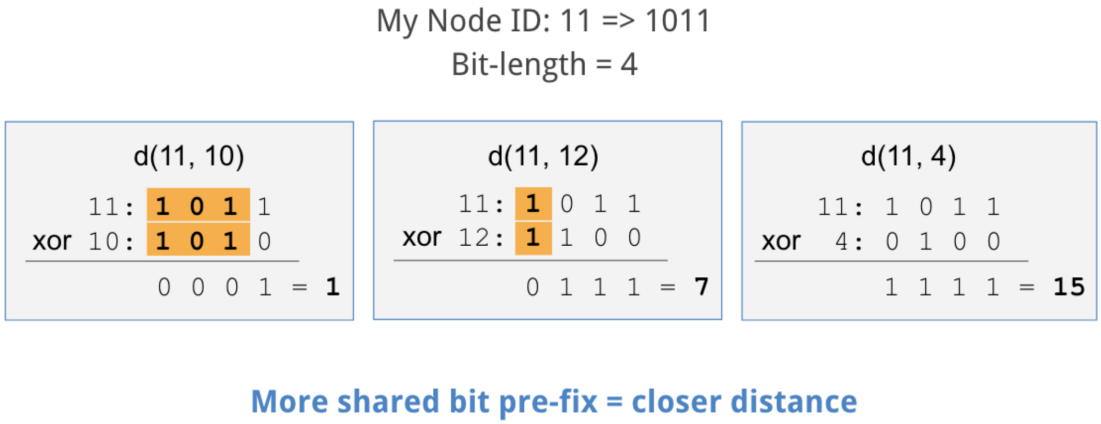
2. It should be *impossible for any individual MN to manipulate or even to predict the future rank ordering of MNs*, since this would allow a

malicious MN to influence the selection process by selectively attacking (say, through a DDOS attack) other MNs in attempt to advantage its own position.

These two criteria appear on the surface to be at odds with one another: how can you have something be deterministic and computable, while still making it impossible to control or predict the result? The Dash project elegantly solves this problem through a very clever idea: by utilizing the random, unpredictable aspects of the mining proof-of-work process, which no participant controls.

The trick is to introduce the concept of an *XOR distance metric*. Essentially, this allows us to take the output of a hash function– say, the hash of the previous block in the blockchain– and compute the "distance" (really, just a number) of this hash to some other hash– say, the hash of each MN's collateral blockchain address (i.e., the blockchain address that contains at least the minimum required coins to be a valid MN). The figure below illustrates how this process works through a simple example:

Figure 4: The XOR Distance Metric (Source: *XORRO P2P* Project)



4.7 The XOR Distance Concept

While this notion of distance is very different from the ordinary definition of distance we are used to seeing in geometry, it is simple to compute and it is consistent: the XOR distance from a hash to itself is zero, and the XOR distance between a hash **A** to some other hash **B** is equal to the distance from **B** to **A** (it also satisfies the so-called triangle inequality). This suggests a good

way to decide the rank ordering of MNs for each new block: simply compute the XOR distance between the hash of the previous block to the hashes of each of the Masternode collateral addresses. Since every functioning node in the network can retrieve a complete and accurate list of the MN addresses at any given time, this is trivial to compute. The MN addresses that turn out to be "closest" to the block hash are then selected by mutual agreement as the next recipients of the MN block reward.

Why should we dwell on the precise technical details of this mechanism? The reason is because it is a fundamental concept that appears again and again in different contexts in the Pastel project. Everyone can agree that there is no way to control the hash of a block on the blockchain: it is a side-effect of the mining process. That is, the miners don't care about the particular nonce, which determines the hash of the block, that they find through mining– they just want to find a nonce that works. Thus, the hash of the previous block in the chain can be considered what cryptographers refer to as a "nothing up my sleeve number."

Like a magician who attempts to "prove" to the audience that there are no cards hidden in his sleeves, using the block hashes that arise from the proof-of-work process in this way "proves" that there is no trickery involved in the rank ordering procedure. Put differently, network participants can rely on the block hash being generated in a "random," uncontrolled way. That is, a system for rank ordering the MNs based on the XOR distance metric will not tend to advantage or disadvantage any particular Masternodes; thus, each MN will end up with the same *expected value* in terms of receiving newly minted coins. Of course, since the system is inherently probabilistic (i.e., the miners are testing out millions of random nonces in order to mine a block), certain MNs may receive a larger portion of the block reward than other MNs over a given time interval. But in the fullness of time, such discrepancies will even out.

4.8 Security Implications of the Rank Ordering Method

The XOR distance metric implies that no network participant can predict what the future rank ordering will be for upcoming blocks, since no one can say what this will be until we first know the hash of the previous block. But, by definition, we won't know what this hash is until the block is actually mined. Furthermore, Bitcoin takes advantage of the so-called *Merkle Tree* concept, in which the hash of all previous blocks is included in each block. This implies that no one can create valid blocks "in advance" and hold these on the sidelines: every time a new block is added to the blockchain, the resulting hash must

be part of the next block, rendering any mining computations done before the new block obsolete. Therefore, any MN is equally likely to assume any rank in the next ordering process. To see why this is so important for the security of the network, consider an analogy to a historical financial instrument known as a *tontine*.

The idea of a tontine is that a group of investors each invests the same amount of money (say, $1,000) at the beginning, and later the entirety of the funds (plus any accrued interest or investment gains) is paid out to the last remaining survivor among the group of investors. Needless to say, this morbid contract generated a fair amount of controversy in its time, even becoming the subject of various Victorian mystery novels. A moment's thought suggests the reason for this: whether or not a person dies is not necessarily a random outcome– the probability can be changed by one investor killing another investor, with each death bringing the murderer close to prize.

While the idea of one Masternode owner physically killing another seems absurd, the digital equivalent of this would be some kind of Internet-based attack (e.g., a *distributed denial-of-service*, or *DDOS*, attack) intended to knock the other Masternode machines off-line, or otherwise interfere with their normal operation, as a means of benefiting the position of the attacking MN. But because of how the MN rank ordering process is designed, a malicious MN would need to target *all* the other Masternodes, rather than being able to focus their attacks on the MNs that are ahead of them in the rank ordering (i.e., only the nodes that are preventing the attacker from being next in line to get the mining reward). Since attacking all of the MNs in this way is likely to be difficult or impossible– at least assuming a significant number of independent MNs active on the network– this is not generally a practical strategy for a rational attacker (i.e., an attacker rationally expecting to profit financially as a result of the attack).

This concept fair rank-ordering procedure is critically important to the functioning of the higher level services built on top of the base blockchain layer provided in the Pastel system, such as the registration of new artworks and the processing of artwork trades. If we always have a secure and reliable way for all the Masternodes to reasonably agree on which of them should be selected for which rewards, and also for selecting which is responsible for providing certain expected services, we can create a scalable system that can intelligently break up the work in a decentralized manner and verify that the required services were carried out accurately through random *peer-to-peer* inspections.

4.9 Economic Role of Masternode Operators

A useful analogy for understanding the potential of the Masternode concept is that of a franchise business, the most iconic example of which is *McDonald's*. The premise of the franchise model is that the creator of the franchise has figured out a superior business plan and set of operating procedures for ensure productive operation of the enterprise– say, a profitable and competently managed restaurant– and is offering potential franchisees this plan, along with all the various things required for the franchisee to get start in business, such as:

- Supplies, such as ingredients, napkins, and containers.
- A well-known brand name, which creates a higher level of initial customer interest.
- A detailed handbook describing each team member in the staff, which of team member is responsible for which tasks, and how each of those tasks should specifically be carried out.
- Marketing and promotional assistance to help franchisee drive and maintain customer engagement levels.
- New cost-saving technologies, such as automated ordering kiosks and smartphone apps.

In return, the franchisee pays the owner of the franchised business in various ways. In the case of McDonald's and other large restaurant companies, this usually comprises a large up-front payment (sometimes in excess of $1 million), together with a revenue royalty: a percentage share in the ongoing gross sales of the restaurant. The franchisee is willing to enter into such an agreement because it allows them to establish themselves in a business in which, if they follow the operating guidelines effectively, they have a reasonably high probability of earning income from the operation of the new restaurant. That is, the value of the assistance and structure given by the franchised business is sufficiently valuable to offset the large and continuing cost of becoming a franchisee. At the same time, although the quality of the franchised business and the expertise and insight embodied in its organization and structure are paramount to the success of the new franchised location, it is not enough by itself: the ability for a franchisee operator to act (to a degree) independently means that the franchised business loses some degree of control in the specific manner in which the business is presented to the public.

So what does this all have to do with Masternodes? If you think about it, the structure of the Masternode "business proposition," at least as outlined in this document, resembles that of the franchised business model. In this analogy, the franchisees are like Masternode operators, who invest an up-front amount of capital (i.e., the coins required to collateralize and operate a Masternode) in return for the opportunity to become part of the network, just as the new store operator gets to be part of the McDonald's network of stores. And just as the new franchisee has a series of obligations to McDonald's– for example, they must serve only the products specifically permitted in each region by the corporate parent, and they must maintain certain minimum standards for cleanliness, sanitation, safety, and security– the new Masternode operator must perform a series of obligations when called upon by the rank ordering process, which include:

- The parsing and validation of submitted blockchain tickets.
- The application of NSFW and duplicate content detection.
- The preparation of the image files for insertion in the off-chain storage system.
- The facilitation of decentralized trading of registered artworks, with the MNs acting as brokers on behalf of users.

Put differently, the Pastel project provides the prospective Masternode owner the opportunity to participate in the rewards and work involved in actually running the network. It gives them the authorization, or "license," to be allowed to be part of the system, and it also provides the tools required for a non-technical user to fulfill all of the actions expected by other users from a properly functioning Masternode. Just as McDonald's explains to the new franchisee the precise manner in which to make the fries–the temperature and duration of frying, the amount of salt to apply, the tool to be used for scooping the fries, etc.– the Pastel software gives the Masternode the software tools required to, for example, automatically determine if a newly submitted candidate artwork is a near duplicate of an existing artwork– a challenging problem that until recently had no good open-source solutions– all without the manual intervention of the Masternode owner, who can essentially "earn money while they sleep" as their computer slavishly follows the procedures encapsulated in the Pastel computer code.

Taking this analogy further, the cost of the raw ingredients and consumables in the McDonald's business model is similar to the role played by miners

and the proof-of-works system used in the Pastel project. Just as McDonald's and the franchisee could theoretically make more money in the short term by jointly deciding to spend less on ingredients–since the cost of this comes straight out of what would otherwise be profit that could be apportioned between them–the Masternodes as a group could be more profitable in terms of return-on-investment if a higher percentage of the overall block rewards (i.e., new coin production in the network) is diverted away from the miners who are providing the proof-of-work computations securing the blockchain and instead sent to the Masternodes.

However, it's obvious that such a decision would be dangerous and could potentially destroy McDonald's overall business model: by sacrificing the very thing that creates value in the first place for the customer–the product quality and reputation–we undermine the future of the brand and company, so that any initial gains in profits are outweighed by the potentially catastrophic loss of confidence in the brand by the public.

Similarly, the level of compute power represented by the set of active miners for a proof-of-work crypto-currency is the fundamental "product" on offer in a potential blockchain project: the reasonable believe by a skeptical third-party observer that transactions on the network are secure and immutable; that double-spending attack risks can be mitigated by waiting for enough confirmation blocks; and that one's coins and "digital assets" are safe from theft as long as users can keep their private keys secure. By sacrificing this level of security– which is implicitly what happens every time a project decreases the total proportion of newly minted coins that goes to miners (e.g., giving miners 50% of the block reward instead of 100%, you incentive the network to produce half as much compute power or hash rate, thus reducing the security of the network)–ultimately the project risks the complete loss of integrity of the whole system.

Continuing the analogy, the role of McDonald's corporate functions, and the previously mentioned revenue royalty that the corporate parent deducts from the sales of a franchisee's restaurant, are similar to the function served by the Pastel Growth Fund, which receives 5% of the total block reward– funds that can be directed, in a decentralized way, by a vote of the Masternode owners as a group, using the integrated voting system. Just as McDonald's reinvests much of these fees back into the business, by developing new products and systems with the ultimate goal of making the franchisee and McDonald's more money and by promoting the business through advertising, marketing, and public relations efforts, the Masternode owners who submit Income Fund expense reimbursement requests for possible voting by the Masternodes can make their best arguments for why a given expense is useful or productive for

the network as a whole; this might include the development of new functionality through updated software; the promotion of the project by getting the coin listed on a new exchange; promotion of the project through the enlistment of well-known or respected digital artists who could register their original works on the system; the purchase of select rare artworks that are available on the network as a means of introducing liquidity and as an incentive for worthy submissions.

Since this 5% is essentially a "forced donation," (just like how the franchisee is legally bound to pay McDonald's their ongoing fees), each Masternode is best served by voting for the payment request that appears to have the highest probability of yielding a positive return for the network, either in the form of increased usage or in the form of market price appreciation of the coin. An important distinction to make, and the reason why the analogy is ultimately too limited, is that the relationship between McDonald's and its franchisees as a group is effectively centralized in nature: McDonald's corporate is the ultimate arbiter and decider of what will happen in the system. Here, there is no central authority that has the same privileged, all-powerful position– instead, authority in the network (e.g., the right to register a given artwork as a MN) is granted, provisionally, and with a readiness to withdraw such authority depending on circumstances (e.g., the Masternode owner sells the coins and no longer has enough collateral to qualify as a valid MN), by participants of the network in an agreed-upon fashion. Here is the key: because every step is always performed by multiple, "randomly" selected, (and presumably, mostly, independent) actors, and because each step in this process is publicly available for all nodes to inspect, these actors can all check each other's work and thereby "keep the network honest," at least assuming a majority of the nodes are honest to begin with.

Unlike network services, such as accurate and timely duplicate detection services by a Masternode, certain work can not happen in an abstract vacuum. Talented and creative developers must be paid, generally in fiat currency, a high amount of money to write the secure and professional code that users reasonably should expect in such a project. The purpose of the Income Fund at its core is to serve as inducement for the efforts of a dedicated core group of contributors. Since this is all done out in the open, in a decentralized way, we believe this method offers the best compromise between centralization (and what is referred to as "scamminess" in the crypto-investing community, particularly in the form of pre-mines or promotional ICOs) and the completely undirected "wild west" approach used in the Bitcoin network, where 100% of the block rewards go to miners of the coin, and all development is done independently by individuals acting in their own capacity, or done at the behest

of endowed foundations whose goal is to advance the state of the technology. Since Bitcoin has progressed so far in valuation, many of these entities have the means to continue this work indefinitely. New projects such as Pastel must find the right balance, which we believe the voting system accomplishes.

5 Artwork Registration Workflow

Before delving into all the details of how the various parts of the Pastel architecture work, we will first provide a simplified walk-through of a typical use case for the network– registering a new artwork. In brief, the process is comprised of the following key steps, beginning with an artist who creates an original digital artwork (in the form of a high-resolution image file) and now wants to register it on the Pastel network:

1. Assuming the artist is new to the project, their first step would be to download the Pastel desktop wallet software for Windows/Mac/Linux, which will be 100% open-source and available for download in multiple places.

2. After installing and launching the wallet software, the artist will generate a wallet address (similar to a Bitcoin address), and will also be prompted to generate a new Pastel ID, which is a public/private key-pair (based on the EdDSA protocol using the curve Ed-521).

    - The Pastel ID will serve as the primary means of identifying the user on the network and for authenticating artworks and other network requests.
    - Note that all network participants, including Masternodes, will be required to generate an Pastel ID and to use this identity in all their interactions on the network, and the ID will also be used as the basis for the reputation tracking system.
    - The creation of a new ID (and the tying of this ID to a particular MN collateral blockchain address, in the case of a Masternode user) is memorialized by the user writing an identity establishing "ticket" to the blockchain, which can be done automatically in the wallet software.
    - The act of writing this ticket will cost the artist some number of coins, since that is how data is written to the blockchain. This cost is important because it protects the network against "spam"

attacks, in which a malicious user could create a huge number of fake tickets in order to overwhelm the system. This implies that the artist must first obtain some coins (say, by purchasing them on an exchange and sending them to their Pastel wallet address) before they can do anything with the network.

- If the artist has a following on social media or maintains a personal website, then they might reasonably post their Pastel ID public key there, along with an explanation to their followers about how they can use this public key to verify that any purported works of art are, in fact, made by that artist. Maintaining the security of the Pastel ID private key is thus paramount, since knowledge of this key would allow any user to impersonate the artist on the network.

3. Once the artist has established a valid Pastel ID, they can begin the process of registering a new artwork. Using a familiar web-like interface, the artist fills in various fields with the information describing their new piece, such as the title of the artwork or the number of rare digital prints to create. In particular, the user would also select the art image file in a process similar to that used by web email providers for attachments. This step is analogous to filling in an application form, and results in the creation of a "ticket" data file that specifies every aspect of the candidate digital artwork, including the file hash of the actual art image file.

4. When the artist is satisfied with the contents of the candidate ticket file, the wallet software can then automatically sign the hash of the ticket file data with the artist's Pastel ID private key, known only to the artist–thus establishing the authenticity and provenance of the new artwork in a cryptographically secure way. The signature also secures the ticket data against potential unauthorized modification by malicious MNs, since any change to the ticket data would result in a different hash, thus invalidating the supplied digital signature–all of which can be verified by any node.

    - To encourage a high "default level" of operational security for users, we introduce a secure and efficient key signing procedure, which takes advantage of the observation that smart phones are much easier to secure than PCs (e.g., the default security settings on the iPhone result in a level of security that is far higher than that found on nearly all computers). In our new system, the user can store a special video (generated by the wallet software when the

ID is generated) in the camera roll of their smart phone; this video displays an animated QR code, which, when displayed to a standard web-cam (included in nearly all available consumer PCs today), transfers the Pastel ID private key to the computer running the wallet software. So long as the initial transfer of the key video to the phone is done securely (e.g., by using Dropbox or iCloud for the transfer along with 2-factor authentication), this acts as a kind of "air-gapped" key– one which cannot be intercepted by key-logging malware; since the private key is never stored on the disk of the PC, it disappears when the wallet software is closed.

5. Here is where the system departs from traditional centralized architectures: normally, there would be some designated URL or IP address pointing to "the server," which would then handle the processing of the file. But in a decentralized system, which machine(s) should the artist be uploading this image file to? How can we decide in a secure but scalable way?

    - The answer is, of course, for the artist to upload the file to the Masternodes. But which one? Again, we rely on the method of rank ordering the MNs in each block discussed in the previous section: the artist connects to the K highest ranked MNs, in rank order. This is important because we cannot allow any user to be able to control which MNs process their requests, since this could be used to attack the network.
    - Thus the artist will now engage with multiple "random" MNs, with a minimum required number of 3 distinct MNs being involved in any processing steps. With the exception of the ID tickets, which can be written by ordinary users, all other valid tickets in the Pastel system must be signed by at least 3 MNs, as well as the parties involved in the ticket (e.g., the artist, or the buyer/seller). Not that these steps are done automatically by the artist's wallet software, without requiring any user intervention other than to confirm amounts or to enter user-provided fields (such as the artwork title). Fields that can be supplied automatically (such as the calculation of valid digital signatures) are provided in the background, with the client software displaying a running status report detailing the registration process to the user as it proceeds.
    - Each of the Masternodes engaged in the registration process then begin a series of steps to validate and to approve or reject the can-

didate image. In this process, all of the various user-supplied text fields, such as the artwork title, are securely checked to ensure that they contain correct/valid information and that they conform to the requirements of the ticket format.

- Throughout this process, communication between users and MNs is facilitated by a novel Masternode messaging system, which is built on top of the *ZeroMQ* and *MessagePack* protocols. This system allows nodes to send each other unencrypted short messages that are guaranteed to be authentic and unmodified in transit, since the hash of each message is signed with the user's Pastel ID private key; in the case of Masternode to Masternode messaging, all messages are also signed with the MN's collateral address private key in addition to the Pastel ID, which proves that all such messages were sent by current valid Masternodes.

6. At this point, depending on the size in megabytes of the candidate art image file and the length of the resulting meta-data ticket that must be written to the blockchain, the MNs will respond to the artist by quoting a registration fee that will compensate the network for the permanent storage and verification of the artwork.

    - Assuming the artist agrees to the registration fee, the artist then sends a non-refundable deposit equal to 10% of the full registration fee to compensate the MNs for running the validation checks on the images (otherwise, a malicious user could tie up the resources of the network by uploading a large number of images that are likely to be rejected as "NSFW content" or as duplicates).

      [QUESTION: Instead of having this non-refundable deposit go to the specific MNs involved in that particular registration, perhaps it should be burned by the user by sending it to a provably unspendable address; alternatively, we could have this address be equal to whatever the address is of the current in-force expense reimbursement ticket. That way, there is no way for bad MNs to scam artists into paying fees to them without actually registering the images.]

7. The MNs involved in the registration process now run the more computationally intensive parts of the verification process: the NSFW check, and the duplicate detection check.

    - The NSFW check is relatively straightforward, using an open-source deep learning model developed by Yahoo. This model, implemented

in TensorFlow and trained on millions of inappropriate images, assigns a score between zero and one for every image provided to it as input. If the resulting score is above a certain threshold, the Masternode rejects the submission as being unacceptable.

- As part of the duplicate image detection algorithm, an image fingerprint vector will be computed for the candidate image. This is a list of 8,064 numbers, detailing the outputs of various pre-trained deep learning models when given the candidate image as input. This fingerprint vector is then compared by the registering MNs against a database of previously computed fingerprint vectors for all of the artworks already registered on the network. If any of these existing fingerprints are deemed to be "too similar" to the candidate image fingerprint, then the candidate image is rejected as being a duplicate.
- Once this image fingerprint vector has been independently computed by at least 3 of the MNs, any new registration attempt for an image whose fingerprint is sufficiently similar (defined in a precise, mathematical way) will be rejected as invalid by network participants for the next N blocks. The purpose of this mechanism is to protect artists against a dishonest Masternode owner, who, rather than registering the artwork registration ticket on the blockchain as expected, instead attempts to register the image themselves, thus taking credit away from the rightful creator.

8. As each of the 3 or more registering Masternodes perform the validation of the image files and ticket data, they communicate the results of these tests via the Masternode messaging system, so that the image fingerprints can be compared to ensure that they all match.

9. After verifying that the candidate image is acceptable for storage in the network, the registering Masternodes will begin the process of converting the raw image file for use in the off-chain storage system, which will be described in detail in a later section. This key step is critical to the overall integrity of the system.

    - Assuming that all of the validation tests are passed by the candidate artwork, each registering MN will then compute a set of digital signatures by signing the hash of the artwork registration ticket data with their Pastel ID private key as well as with the private key associated with their MN collateral address. This serves as proof that

the artwork has been reviewed by a random subset of network participants with "skin in the game"–a vested economic interest in the success of the network; unless an attacker has managed to compromise a majority of the Masternodes (which would mean purchasing or otherwise acquiring a huge number of coins), this step goes a long way in insulating the network from malicious content.

10. Finally, once a minimum of 3 of the registering MNs, including the primary registering Masternode–that is, the MN with the highest rank in the MN rank ordering that participated in this registration request– have signed the ticket with their keys, the primary registering Masternode will then write this ticket file along with the set of signatures to the Pastel blockchain. This is done by means of the *Pay to Fake Multisig* approach described in the 2017 paper *Data Insertion in Bitcoin's Blockchain* by Sward. Essentially, this treats the underlying blockchain as a "write only" hard drive; arbitrary files can be written as a series of transactions, with the resulting *transaction ID* (TXID) serving the role of a filename. Any user of the network can look up this transaction (say, in a block explorer) and parse the result to reconstruct the file. In our case, the file is a simple ticket containing the various pieces of metadata:

    [INSERT LIST OF METADATA FIELDS HERE]

    - Because storage space in the blockchain itself is so limited, it is imperative that we reduce the amount of data that needs to be stored in this manner to an absolute minimum; otherwise, the network will be quickly faced with insurmountable scaling difficulties. Although the primary way in which we combat this problem is by moving the storage of the actual image files to the "off-chain" file storage layer, Pastel also introduces a novel method to increase the capacity of the blockchain for storing the required ticket data through the use of the innovative *Z-Standard* lossless compression algorithm developed as an open-source project by *Facebook*.
    - Z-Standard features a "tunable" compression level, the highest of which (level 22) results in an extremely high compression rate– as much as 12-to-1, compared to a typical maximum of 8-to-1 for common compression algorithms–at the cost of sharply increased computational and memory requirements for the compression process. Still, this is an easy compromise to make, since compute power and memory are fairly cheap and getting cheaper over time in real terms

as technology improves, whereas storage capacity on the blockchain only becomes more precious and limited over time. In addition, Z-Standard allows for the optional use of a *compression dictionary* file; if there is any redundancy of the compressed data relative to the data contained in the compression dictionary, then we can use this to save space by incorporating duplicate data as a reference to the dictionary– we simply create a compression dictionary file using a collection of sample tickets. If we periodically replace this compression dictionary file over time as the network evolves, so that it contains all of the past tickets registered on the network, the storage density should improve over time, similar to how cloud storage services such as Dropbox use file *de-duplication* to reduce the need for storage by eliminating any redundant files stored in the system.

11. At this point, the first half of the process is complete: there is a valid art registration ticket file written to the blockchain (which can be referenced by any node by querying for the *transaction ID* associated with the file storage operation for that particular ticket) that has been signed by everyone– that is, by the artist as well as as by a group of MNs selected in accordance with their rank ordering. But this ticket is not yet "active" on the network.

    - In order to activate the ticket and consummate the transaction, the submitting artist must quickly (e.g., within 3 blocks after the registration ticket is written) send an appropriate portion of the art registration fee to the specified blockchain addresses for each of the registering MNs. Again, this can be handled automatically by the wallet software without the artist having to do anything manually other than grant approval.

12. The purpose of breaking up the registration process into two parts, so that it has a separate activation step (i.e., some condition that can be independently verified by any network participant– in this case, whether or not a certain blockchain address sent a certain number of coins to another given address) is to address the challenging problem of *escrow* in a decentralized system. That is, how can we prevent a dishonest MN from taking the registration fee without doing the work to verify and store the submitted artwork?

    - Although a reputation tracking system could theoretically penalize the dishonest MN (indeed, we plan to employ such a system in Pas-

tel as an additional safeguard, as described below), the unfortunate artist would still be out of their registration fee– after all, who individually could "make the artist whole," without digging into their own pocket, if transactions are necessarily immutable?

- While there are implementation of escrow in Bitcoin transactions, these are not true escrows in the normal sense, in that users cannot recover escrowed funds in the event of non-performance by the other side: all a user can do is ensure that the other side cannot get his funds if they do not perform, which, while useful in terms of incentives, is still a disaster from the standpoint of the user who has also lost access to his funds.
- Our design completely side-steps this performance risk: the other side has already written the ticket to the blockchain in a verifiable way at the point when the user must pay the fee. Put differently, our design forces the MNs to commit to a certain course of action by permanently writing this to the blockchain, such that, once the required fees have been sent–which again, no one can dispute, since this is easily checked by all participants using the blockchain–the specified course of action is automatically set into motion. This strategy exemplifies one of the guiding design principles of the project: rather than relying solely on a reactive system that must detect and punish bad behavior on the part of network participants, we always prefer a system that eliminates the opportunity for dishonest behavior completely; this sentiment is analogous to the well-known software development and security precept that "the best (most secure) code is the code that doesn't exist because it is not required".

13. Once these transactions have been confirmed by the network, the artwork is activated and is now available for trading with other users, who can view a preview of the artwork by searching for the specific profile for that artist, or by searching for the specific transaction ID (say, because the artist posted about the new work along with the transaction id on social media), or by using the wallet software to browse a list of "trending" artworks that are notable in some way (e.g., because they have experienced a large number of views/purchases/trades or a significant recent price change; such statistics are collected and maintained by the MNs in a decentralized way).

6 Artwork Collecting and Trading Workflow

We have described a typical process from the perspective of the artist who is trying to register new artwork. What about from the perspective of the fan or collector? There are three typical scenarios to deal with:

6.1 The Types of Users

- A collector is interested in a particular work of art or artist that the collector is already familiar with. For example, suppose that the collector follows the artist on social media, and the artist posts a picture of the artwork and includes a text string in the post (the transaction ID) that the collector can search for in the Pastel wallet software. Alternatively, the collector can search for the artist's Pastel ID public key, which will display a list of registered works by that artist.

- A collector is generally interested in purchasing artwork on the system, but does not have a particular artist in mind; instead, the collector prefers to browse the offerings on the network, perhaps sorting by various tags or fields in the artwork (e.g., searching for a particular string of text in the title field). The challenge with any kind of browsing is that it threatens the decentralization of the system. We propose a system in which the Masternodes can maintain basic statistics about the network and the registered artworks (e.g., the number of times a given artwork or artist has been searched for; the number of times an artwork has been purchased in the past 24 hours; the total implied change in value of all of an artist's works registered on the network over some period); these statistics are used to generate lists of "notable" or "trending" artwork, which users can use to filter the available content.

- A collector is not looking for a particular artwork or artist, but has a specific interest in mind; for example, one collector could be interested in contemporary fine art, while another collector is interested primarily in anime artwork. For this application, users will be able to create and distribute custom *curation lists*. These lists act as a filter across all the valid registered works in the system, and can serve the role of a "quality control" enforcer, highlighting just the works that a user is interested in. Since the user can switch between such lists at will, and all of the lists are voluntary (i.e., a user is free to see all registered works in the network regardless of origin), this method offers the best compromise between user experience on the one hand and censorship/centralization

on the other. It also creates a new potential role in the crypto-economic ecosystem of the network, where skilled curators add value through their ability to select works that appeal to particular groups of collectors (future functionality may include an incentive scheme system, similar to an affiliate marketing relationship, in which curators receive a share of the proceeds of artworks purchased as a result of their curation).

6.2 Associated Workflow and Integrated Trading Functionality

In all these scenarios, the resulting user experience is ultimately similar: a rich visual display of artwork thumbnail previews is presented in the wallet software; each thumbnail can be selected to reveal all the information associated with that artwork, such as the quantity in circulation, the artist's name, the price history of the artwork, etc.). All of this metadata, as well as the image thumbnail itself, is supplied to the wallet software by a group of Masternodes; these responses are compared to confirm the correctness and to protect the networks against malicious Masternode operators while remaining decentralized.

In particular, the user can see the *order book* for the artwork, along with buttons to buy or sell the work. Obviously, the sell button will only be selectable if the collector already owns that particular artwork. In the order book display, the collector can see how many copies of the artwork are available for sale, and can see if the seller at a certain price is the artist or another collector. Each of the available sell orders is the result of a *trade ticket*, an important concept in the Pastel system. If the collector has a sufficiently high balance of coins available in their wallet, then they can click the buy button to initiate the process of purchasing the artwork. The user simply has to specify the price (in the native currency of the network) and the desired quantity; this information is used to generate a short ticket, which the collector then signs with their Pastel ID private key.

The executed trade ticket is then submitted to the Masternode network in a similar way to how the artwork registration ticket is handled: the highest ranked Masternodes for the current block are contacted via the messaging system. They independently validate the contents of the trade ticket, ensuring that the trade refers to a valid, existing artwork on the network, that the price field is valid (e.g., not a negative number), and that the buyer actually has the required number of coins to fulfill the trade. These Masternodes then sign the trade ticket with their Pastel ID and their MN collateral addresses, and

when enough of these Masternode signatures have been collected, the highest ranked registering MN then writes the ticket to the blockchain in the same way as the artwork metadata ticket is written.

In order for a trade to be consummated, the trade ticket just written must match another valid trade ticket already in the system that takes "the other side of the trade"–for instance, offering to sell for a price less than or equal to the price specified by the buyer. Assuming such a match is found, the buyer is then presented with a confirmation screen which displays the Pastel ID of the seller and the blockchain address where the coins should be sent to pay for the purchase. Because both the "buy ticket" and "sell ticket" are already written to the blockchain, we can avoid the escrow dilemma again in favor of a trustless system. Essentially, once the required funds are sent by the buyer, the Masternodes will then be selected (again, using the same rank ordering selection criteria) to write a *trade confirmation ticket* to the blockchain that memorializes the trade. The set of all confirmation tickets is used by each node of the network to compute which Pastel ID owns which artworks (along with when the artwork was purchased as well as the price).

Effectively, what we are really doing here is using the classic Bitcoin style blockchain as "dumb storage" for tickets: it is a first layer of trust, but not the final level of truth. In addition to the raw data being written to the blockchain in the form of special transactions–one for each ticket written–each Masternode then has the task of parsing each of these tickets, determining which of them are valid (e.g., by checking the various fields of the tickets, or by verifying the digital signatures used to sign each ticket), and then computing the implications of each valid ticket in order to construct a coherent and consistent "world view" of the state of the system: which works are available, and "who" owns which artwork (and in which quantity), and who should be authorized to request original image files from the off-chain storage system.

Once the trade confirmation ticket is written to the blockchain successfully, the collector now "owns" the artwork and can control. In particular, the collector can now request from the Masternodes via the messaging system (signing the request with their Pastel ID, which can be verified to own the requested artwork by the Masternodes directly from the blockchain) to download the original, high resolution art image file. This request might take a couple minutes to process given the way that files are broken into chunks and distributed throughout the network for off-chain storage, but eventually the Masternodes will send enough chunks back to the user's wallet software to allow the file to be reconstructed.

6.3 Piracy and Unauthorized Usage

Of course, at this point, nothing stops the collector from taking this rare digital artwork and sharing the full resolution "original" file on file-sharing websites such as *The Pirate Bay*. We do not attempt to restrict this (because that would be impossible), but point out that it shouldn't really matter anyway, just as someone else possessing a PDF file of Spider-Man number one doesn't detract from the value of a genuine original edition of that same comic book. What matters is that, unlike the people who downloaded the full resolution image file in an "unauthorized" manner from a file sharing site, the owner of a bona-fide rare digital artwork can *prove* ownership. That is, the collector can prove that he or she supported a specific artist at a specific time. Furthermore, the actual owner of a rare digital artwork can sell it again later to another user.

6.4 End User Experience for Collectors

Once a collector has purchased an artwork from an artist or as a resale from another collector, the purchased artwork then appears in a section of the wallet software called *My Rare Art Portfolio*. The user can customize the presentation (e.g., position, size, etc.) of the artworks in the portfolio, and can generate a URL to share this on a "view only" basis to other users. The collector can click on any of the owned artworks in this section and see a screen listing all the information pertaining to that work; in particular, the collector can monitor recent trades in that same artwork, which can be used to get an estimate of the current potential value. The user will be able to see aggregations of this price data information, which offer an estimated summary view of the value of the collector's rare art portfolio, how this value is changing over time , and which works are the greatest contributors to these changes. By sharing and analyzing this price and portfolio data across the network, various opportunities for "gamification" become apparent, with collectors competing against each other for the largest percentage increase in portfolio value over time, thus increasing user engagement and usage of the network.

7 Initial Distribution and Emission Schedule

There are only a few basic ways to start a new proof-of-work based cryptocurrency, and these vary greatly in their degree of decentralization and incentive structure. The "purest" way is to do it the way Bitcoin started, which we can call the "fresh start" approach: you publicly announce availability of

the open-source software, so that anyone who is interested can download and install it and run a node (or even a Masternode if they own enough coins), and anyone can mine it by running the proof-of-work process on their machine. The single most important factor in this distribution is the presence of a *pre-mine*, which is where the developers of the new coin privately mine the coin with no competition before making the software available to the public. Historically, the early crypto investing community has taken a dim view of pre-mines, with the arguments in favor of them (e.g., they create an incentive for the developers to create and maintain the system) outweighed by the criticisms: they greatly reduce the degree of decentralization in the network and create a dangerous situation for later buyers of the coin, as the risk of "whales" dumping huge numbers of pre-mined coins is always present, even if it is not readily apparent to the public.

Another way to distribute coins is to essentially pre-mine the entire supply, or a large portion of the ultimate circulating supply, and then distribute these coins, either through an *air-drop*, in which they are given away for free, or by selling the coins to investors, usually in the form of an "initial coin offer", or *ICO*. We believe both of these methods are critically flawed:

- **Air-Drops** create various issues; most serious among these is how to ensure that coins are in fact going to the intended recipients–genuine new users of the system, each of whom receive only a small amount of coins, so as to increase the number of potential participants– and not to malicious actors attempting to exploit the air-drop for selfish ends. Most attempts at "fair air"-drops– for example, so-called "faucet" websites that automatically send a small amount of coin free of charge to anonymous users who submit a request, or the attempts by the *Stellar* project to tie the air-drop transaction to a specific *Facebook* profile–are routinely undermined and subverted by hackers, either by mimicking the appearance of hundreds of users (say, by using a series of VPNs to change the originating IP address) or by using other forms of clever social engineering; an example of the latter would be the use of Amazon's *Mechanical Turk* system, which permits users to outsource a task to a group of anonymous low-paid workers, to harvest a large number of valid Facebook profiles in ordering to receive a disproportionate share of the air-dropped *Lumens*– an attack that was infamously carried out by Rhett Creighton, thus exposing the security risks inherent in such an approach. In addition, air-dropped project necessarily begin their lives in a totally centralized fashion, with the whims of the insiders controlling how, when, and to whom the coins are distributed.

- **ICOs**, while effective at raising large amounts of funds, present a project with a host of issues from an incentive structure and decentralization standpoint. Most critically, they create the incentive for exaggerated claims of utility and market potential, so that the promoters can recoup the large costs of marketing the ICO transaction, which often exceed millions of dollars. Perhaps worst of all, they get the incentive structure exactly opposed to what you would want from a game-theoretic standpoint: since the promoters get the money up-front, they lose much of the sense of urgency required to effectively develop the software. Many projects which "ICO'ed" over 1.5 or 2 years ago have yet to release working versions of the software, or even to show their progress by making their code base open-source so that others can track the pace and direction of development. Much of the funds raised in the ICO process are spent wastefully, or in ways that do not increase the actual utility value of the project. Expensive banner ads on high-traffic crypto-currency related websites, and hordes of paid marketing shills on social media and chat services such as Telegram and Reddit, are the norm.

  When the ICO proceeds of such projects are eventually exhausted, many of them will no doubt cease development, leaving such projects in a state of limbo. In addition, ICOs create a host of regulatory and legal problems, with the vast majority of ICOs now requiring a large amount of paperwork and compliance for residents of Western countries such as the United States. Put simply, many such ICO promoters are breaking the law by conducting unregistered offerings that would be considered securities and thus subject to all sorts of additional limitations and requirements. Similar to air-drops, ICOs also require that the coin start out in a totally centralized way, with the developers controlling every aspect of the distribution and generation of coins. Not surprisingly, many of these project have later proven to possess are much larger degree of centralization than initially promised to investors; for example, security issues in projects such as *Bancor* have led to supposedly "immutable" transactions being reversed or canceled by central authorities (i.e., the developers) of the network.

  From a security and decentralization standpoint, the fresh-start approach, where a new coin is announced and "mineable" from day one, is clearly the best of these options. The only way for a user of the network to gather a disproportionate amount of coins in such a system, of which Bitcoin is the best example, would be for the user to add much more compute resources to the system than anyone else, thus increasing the

security of the network. While early miners will tend to generate a large number of coins compared to later users, given the exponentially declining mining block reward, there is no way for network participants to have an outsized effect on the network without continually supplying proof-of-work, which expends real-world resources such as electricity, to the network.

The problem with the fresh-start approach is that, if you use a similar coin inflation schedule to that of Bitcoin, it takes a couple years for there to be enough coins in existence in order for the project to have a large enough market capitalization to somehow–directly or indirectly–financially support the efforts of the development team. In the case of Bitcoin, which, being the first workable crypto-currency design, attracted a large amount of "free" development effort from independent software developers around the world–many of whom are high proficient coders–this drawback did not prove to be a large detractor to the success of the project. Future projects, however, must grapple with this problem.

7.1 Developer Compensation

The issue of how to compensate the developers of a new decentralized project is complex. Most approaches introduce a large measure of centralization, such as the approach taken by the *Z-Cash* project, where 20% of the mining block reward is automatically sent to a special address controlled by the developers. Although no one can fault the developers of Z-Cash for adopting the structure that they did, since they are free to develop and release whatever system they want, such a scheme creates a serious bottleneck in the process of eventually making the project decentralized. For example, suppose for some reason that the Z-Cash developers abandoned their project; while their portion of the block reward makes this unlikely (since they are paid in coins and thus have an incentive to develop the project), suppose hypothetically that they were forced to abandon it for other reasons, such as living the team living in a country which made their participation in the project illegal.

Who would then be able to "carry on the torch" to develop the project? Why should any outsider want to get involved, since there is no clean way for a new developer to receive such compensation from the network in a manner similar to that of the original Z-Cash team? Furthermore, once the original development team ceases work, their receipt in perpetuity

of a large portion of the block rewards presents a fundamental economic conundrum from the project going forward; if the insiders choose to continue to liquidate their share of the block reward despite no longer contributing to the development of the system, it creates a form of taxation and produces a "dead-weight loss" economically; the market, in the form of the various centralized exchanges on which the coin is listed, would be forced to absorb such insider sales, making it harder to accomplish the sustained price increases needed to allow decentralized project to flourish through persistent interest from new investors of the coin or users of the coin's network.

7.2 The Fork Mechanism

There is another option to distribute a new crypto-currency project: the *fork*. In a fork event, a *snapshot* is taken of a crypto-currency project (known as the "parent" project or blockchain) as of a specific block number, with the balances of coins at each address crystallized at a given point in time. These balances, which are known as the set of unspent transaction outputs (*UTXO* for short), are then automatically credited to those same addresses in a brand-new coin project. The most famous example thus far of a fork is the *Bitcoin Cash* project, which was distributed as a fork of Bitcoin to holders of Bitcoin as of May 15th, 2017. Essentially, this meant that Bitcoin users, without doing anything other than possessing the private key associated with their coins (i.e., if the users kept their coins in a software/hardware/paper wallet, as opposed to on an exchange or service such as *Coinbase*), would be credited with the same number of Bitcoin Cash coins that they had in each wallet address. One way to think about this process is that the developer of the forked coin is able to take all of the private keys on the network, and, without ever learning what those keys are, create a set of identical "locks" for the new coin that are opened by the same key as the corresponding coins in the original network.

Forks are interesting because they present a hybrid mechanism, where they inherit many of the security properties of their "parent" coin. That is to say, if the parent project was fairly distributed, with no pre-mine, and the coin distribution consisted of large numbers of widely dispersed users without an excessive degree of ownership concentration (e.g., a single user owning 60% of all the coins), then the forked coin will also begin life with this same "fair" distribution.

Importantly, forks allow a new project to skip the long and tedious time period at the beginning of a new network in the fresh-start approach, where there are so few coins outstanding that the market value of the project is tiny from the outset. Instead, the network begins with a large number of users. If such a fork is advertised early on in its development, so that investors and users are free to acquire the parent coin on an exchange or through mining, this leads to an efficient, market-based allocation of forked coins. That is, users of the parent coin who don't believe in the forked coin's prospects can quickly sell the forked coins that they receive. Potential investors of the forked coin might care more about the forked coin than the parent coin, acquiring the parent coin before the snapshot date with the intention of participating in the fork event.

The fork concept presents an alternative financing mechanism that retains a fair degree of decentralization. In this mechanism, which is what the Pastel project is using for distribution, the developers of the project find a suitable project to act as the parent coin in a fork. What makes one project more attractive than another as a "fork candidate"? We would propose the following attributes:

1. The parent coin should be a true proof-of-work project (in the Bitcoin mold) with zero pre-mine. Ideally, the coin should have been actively mined by a large number of independent user over an extended time period. The more similar to Bitcoin the parent coin is, especially in terms of the underlying code base, the better; the Bitcoin code base is the best understood and by far the most secure and well-tested crypto-currency project– starting from another code base would create a large degree of uncertainty as to the underlying security of the parent chain, which could create a risk for the new project before the fork event is consummated. The fork development team should verify that the network functionality is still intact, and that there are no known security issues with the project before proceeding.
2. The parent coin should essentially be abandoned, without an active development team that might be hostile towards newcomers and view the new team as attempting to undermine their influence or even "hijack" their project. Such infighting starts the network out on a negative note, and creates uncertainty and doubt in the minds of potential users and buyers of the coin. It could even present

critical consensus issues, with the networking breaking down into 2 competing (and incompatible) versions of the same ledger (confusingly also referred to as a fork). This allows the new development team to acquire the web domains and social media accounts required to effectively support the coin.

   - An important ancillary effect of this requirement is that, in the very beginning stages of a fork, the parent coin can be acquired at low cost in large quantities by the founding development team. So long as the fork event is quickly publicized, with some kind of road-map for users explaining the process, this is nothing more than the result of supply and demand in a marketplace.
   - Anyone who wants to can purchase the parent coins and get the forked coins, and the creators of the fork project can't control this (or if they attempt to, this will be obvious from a basic inspection of the open-source code base for the project, causing the team and project to suffer catastrophic reputational damage).
   - The potentially "excessive" price returns for such early-adopters of the fork project, who is able to purchase the coins inexpensively, must be view through the lens of the excessive risk undertaken by such an early-adopter. If the fork project turns out to not be viable, for lack of development expertise or financial limitations of the insiders, the early adopter is faced with an illiquid market and the loss of their entire investment if they want to get sell their coins to get liquidity.

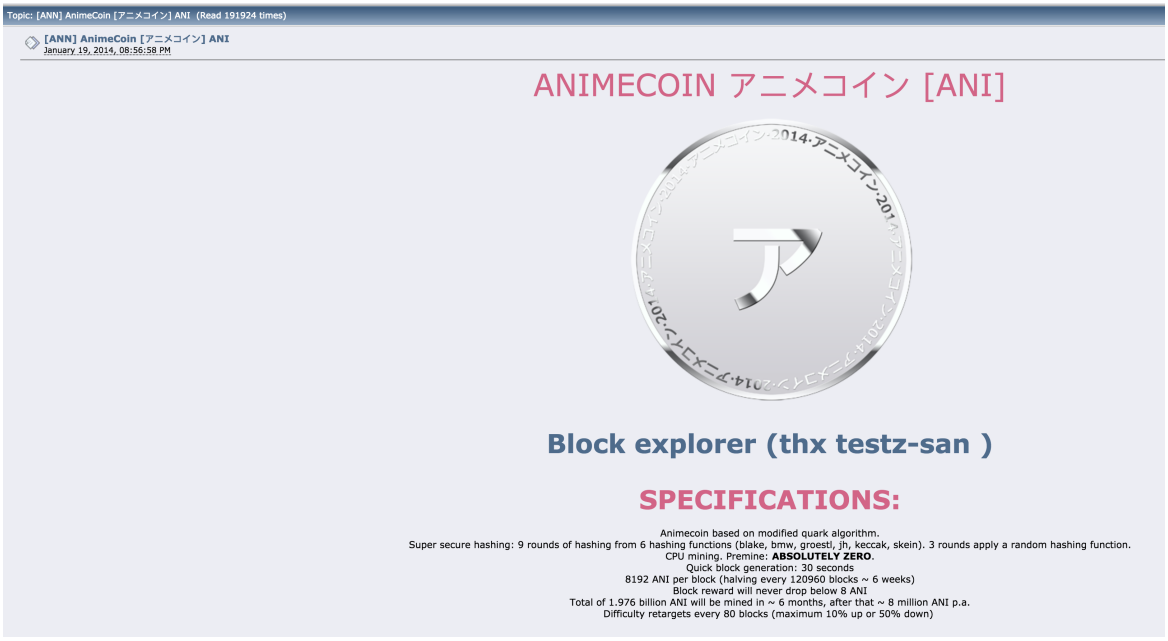
3. The parent coin should have at least one exchange listing with a legitimate centralized exchange, to facilitate outsiders taking an interest in the project before the fork date. Such an exchange listing is a critical link between the "real world" outside of the project. It also greatly facilitates the distribution of the forked coin to non-technical users. Many users of exchanges have little to no ability to manage the process of transferring their private keys for us in the fork coin. By working directly with the exchange site, the fork development team can make it easy for users to purchase the parent coin for Bitcoin and then automatically receive the forked coin, which would also be listed on the same exchange site.

We would humbly submit that the *Animecoin* project, which will serve

as the parent coin for the Pastel fork, possesses each of these desired attributes. Animecoin, also known by the symbol ANI, was originally announced on the well-known *bitcointalk.org* web forum on January 19th, 2014– fairly early compared to the vast majority of the roughly 2,000 crypto-currencies in the market today. Animecoin was intended as currency "for the anime community"; although this questionable goal was never reached (why would they need their own coin instead of Bitcoin? What functionality did it provide?), the coin was nonetheless enthusiastically adopted by hundreds or even thousands of users, many of whom were computer game fans who simultaneously had an interest in crypto-currency and anime as well as a powerful GPU for use in 3D games. As a result, the coin was widely distributed over an extended time period.

While a detailed description of the technical aspects of ANI are beyond the scope of this document, the Animecoin software is essentially a Bitcoin clone (as of late 2013) with a 30-second block time instead of Bitcoin's 10-minute block time, and using the "modified Quark" hashing algorithm instead of *SHA-256* as in Bitcoin. The following figure shows this initial announcement page, and readers are encouraged to read the full announcement themselves for more details and to draw their own conclusions.

Figure 5: The Animecoin Announcement Page (Source: *bitcointalk.org* forum)



Importantly, approximately 2.1 billion ANI have been created through the mining process (these and other statistics can be independently verified by readers by downloading and syncing the open-source ANI wallet software.) However, it is important to understand how ANI was distributed historically: unlike Bitcoin, ANI had an accelerated mining period during the first 6 months of the project after launch, in which the majority (approximately 1.9 billion ANI) were mined, after which the inflation rate of the project was reduced significantly. At present, the ANI network is comprised of nearly 4.5 million blocks, and each new block, which is generated on average every 30 seconds, now produces just 8 ANI as a mining reward. Thus the current inflation rate of ANI is quite low at under 1% per year. Many of these initially minted ANI may be permanently lost as a result of lost computers, deleted/reformatted hard disks (a common activity among gamers), and forgotten private keys.

Although the presence of an open market, in the form of an exchange site such as *Cryptopia*, which is where ANI is currently listed, is critical for the economic underpinnings of a given crypto-currency (this is so because it is what allows network participants to convert their coins for "real world" resources, such as Bitcoin; without this, artists would have no way to recover the registration fees from submitting their artwork, let alone to make a profit through the sale of their artwork. At the same time, it creates a structural risk for the team and early-adopters of the forked coin, in that an apparently stable market, in which supply and demand for the coin (either the parent coin or the forked coin) is in equilibrium, can be completely undermined by the introduction of a "legacy whale" who owns hundreds of millions of coins and suddenly decides to dump these on the market, thus collapsing prices for everyone.

In order to deal with this problem, we propose a modified fork procedure in which distribution of the forked coins is not automatic: rather, holders of the parent coin who are interested in participating in the fork are required to signal their desire to participate by sending a tiny amount of ANI (say, a single ANI, worth under a penny) to a provably unspendable address for each ANI address that they control before a certain block number on a given date (publicized in advance). For users who keep their coins on Cryptopia, we will work with the exchange to automatically handle this procedure on behalf of holders of ANI. In this way, the project will get a confirmation of the true number of "marketable" existing coins in the fork.

Any coins that have been lost through dead hard drives and other causes

will thus not receive an allotment of the new Pastel coin. More importantly, old users of ANI, some of whom might have large balances, but who have not kept on top of developments with the project, would also be left out. Although this is arguably unfair to such unwitting users, who would normally participate in a fork even if they didn't know the fork was happening, not taking any such precautions creates an uncomfortable amount of risk to early adopters. Suppose such a "legacy whale" were to appear, who only recently found out that the old "worthless" coins are now worth thousands of dollars. If such a user is being economically rational, they should not cause the price to plunge by indiscriminately selling as quickly as possible, since this ultimately reduces the value of their own coins. However, such irrational behavior is commonplace in practice, and seems sufficiently grave so as reasonably lead to the compromise outlined above (i.e., the requirement for a contemporaneous ANI transaction to signal intent to participate in the fork).

7.3 Coin Distribution Post Fork

There are several ways to control the inflation rate (also known as the *emission schedule*) of a forked coin after the fork. We have selected a particularly simple method that attempts to follow Bitcoin and Z-Cash as closely as possible. In short, holders of ANI who signal their interest in participating in the fork by sending 1.0 ANI to a specified, unspendable Animecoin blockchain address from all their coin balance addresses (done automatically on their behalf if the coins are held on Cryptopia), before a particular block number occurs (i.e., at the publicized fork snapshot date) will receive a certain number of Pastel coins in return for each ANI contained in any wallet address that sent the 1.0 ANI signaling payment.

To facilitate art transactions that do not result in a comically small number of coins, we have elected to model the new Pastel coin distribution on Bitcoin, but instead of using 21 million as the total, we instead use 21 billion, at some slight loss in expressive power in that each resulting coin will not have as many decimal places of precision as in Bitcoin. This way, a user can talk about purchasing an artwork for hundreds or thousands of coins, instead of 0.002 coins or similarly inconvenient numbers.

As of the fork date, the initial coin supply will be deemed to be half-way distributed, so that 10.5 billion coins will be outstanding on day one; these coins will be proportionately owned by all the ANI holders who

signaled interest in participating in the fork event. Since we will know the precise number of qualifying coins that will participate in the fork, we can compute the appropriate ratio to apply in crediting each balance with the new Pastel coins.

From then on, the coin emission schedule will be set to match that of Z-Cash, which is closely modeled on the Bitcoin emission schedule, as described in more detail in a later section. For comparison, Bitcoin block reward began at 50 BTC per block, with a new block every 10 minutes. Because the Pastel max supply is 1,000x the size of the BTC max supply, we must multiply this by 1,000, but since there is a new Pastel block every 2.5 minutes instead of every 10 minutes for BTC, we must divide by 4; this implies an initial block reward of (50*1,000)/4, or 12,500 Pastel per block. This block reward would be reduced by 50% every 4 years, similar to how it works in Bitcoin.

8 Core Architecture

From the beginning, the primary design inspiration for Pastel has been Bitcoin, which strikes an ideal balance between security and functionality. While Turing complete blockchain projects such as *Ethereum* obviously have more built-in functionality, this expressive power come at a great cost in terms of security and reliability. The fact that one of the core developers of Ethereum managed to lose hundreds of millions of dollars worth of coins because of a software bug in a multi-signature wallet shows that such a system is too complicated to reason about effectively even for the most advanced users.

Just as Bitcoin focuses exclusively on the narrow problem of how to make ideal "e-gold," and optimizes for this single use case in the simplest way possible, our goal with Pastel is to build only the functionality required to efficiently and securely provide the services we need for the specific use case of rare digital artwork on the blockchain, rather than being all things to all people and use cases. At the same time, many of the techniques and approaches we use for rare artwork would apply equally well to other types of files, which we discuss further in a later section.

As such, we wanted our base layer– the foundation on which our project is built–to be Bitcoin. By using a familiar, well-tested foundation, we can ensure that the most basic functions of a crypto-currency network–the secure transfer of coins among users in an immutable ledger–is beyond reproach. However, rather that use Bitcoin Core as the starting point for the Pastel code base, we have instead selected to build on top of *Z-Cash*, which demands an explanation.

8.1 Z-Cash

The Z-Cash project is a direct fork of the Bitcoin code-base (albeit an outdated fork) which makes only very modest changes to the key parameters of Bitcoin:

1. **Proof-of-work algorithm:** Z-Cash uses *EquiHash*, rather than the *SHA-256* hash algorithm used in Bitcoin.

2. **Average Block Time:** Z-Cash has a 2.5 minute block time, versus 10 minutes for Bitcoin.

3. **Emission Schedule:** Z-Cash makes minor changes to the initial rate of distribution of coins early in the network's evolution, but essentially sticks to the same 21 million coins and distribution schedule as in Bitcoin.

4. **Difficulty Retargeting Algorithm:** Z-Cash uses a different mechanism for adjusting the difficulty of the proof-of-work in response to observed changes in the network's hash rate designed to be more stable.

But the primary change made by Z-Cash is the introduction of *Z-SNARKS*, a powerful cryptographic scheme for facilitating on-chain transactions (known as *shielded transactions* that are nevertheless invisible to other network participants, who cannot determine the identities or amounts involved in such transactions. Although the cryptography involved in Z-SNARKS is relatively new, the method has been extensively peer reviewed, and is based on proven cryptographic concepts.

While a discussion of how Z-SNARKS work is beyond the scope of this document, we will briefly remark that we are employing the same "Trusted Setup", or initial set of security parameters, as the Z-Cash project uses. The key generation ceremony used for Z-Cash has been extensively documented elsewhere (See X, Y), and a detailed review of the steps taken should convince even a skeptical observer that the methodology employed was sufficiently rigorous to ensure the overall security of the scheme.

In addition, the development team of the Z-Cash project–which includes several of the researchers who invented the cryptographic methods underpinning the project– are some of the best in the space. The level of professionalism in the development of the Z-Cash project, which was financed by a well capitalized, venture-backed private company (the *Z-Cash Electric Coin Company*, and especially the project's extensive use of reputable third-party security auditing firms which have written detailed publicly available reports certifying the soundness of the architecture, gives the Z-Cash code base a level of security

that is high relative to that of Bitcoin. Combined with the additional functionality from the introduction of shielded transactions, we concluded that Z-Cash was a better starting point for the Pastel code base, with minimal downside.

All of the professionalism in the development of Z-Cash did not come cheap: a whopping 20% of the mining block reward in Z-Cash is "hard-wired" in the software to be allocated to a specific address, which is a multi-signature wallet controlled by the Z-Coin company. This decision, which is justifiable on economic grounds, brings up important questions about the true degree of decentralization is such a network. In the Pastel architecture, we have removed this functionality (similar to the previous Z-Cash fork, *Z-Classic*). Instead, compensation for those who contribute in various ways to the development and and promotion of the Pastel network will be conducted using a decentralized voting system controlled by the Masternodes, as described in more detail below.

8.2 Dash

Of course, one obvious limitation of Z-Cash is that it has no support for the Masternode concept, which is a cornerstone of the Pastel architecture. Thus, the next key layer of Pastel's architecture is the core Masternode code from the *Dash* project, which pioneered the idea in its 2014 introduction. This code, which, like the Bitcoin and Z-Cash code, is primarily written in the *C++* language, was (after being slightly simplified and modified) carefully transplanted into the Z-Cash code base, thus producing a novel combination: a Masternode based crypto-currency with full support for Z-SNARKS. Although there are dozens of Masternode crypto-currency projects based on the Dash code base, most of these are close derivatives or "clones", and none offer the kind of "step change" increase in functionality at the level of shielded transactions.

However, DASH contained many elements that were complex and unnecessary for the functionality of the Pastel project. For example, Dash services such as *InstantSend* and the questionable *PrivateSend* were completely removed, leaving the core skeleton of the Masternode architecture: the selection of the "winning" Masternodes that will receive the next share of the block reward, and the essential code for checking that Masternodes control the required number of coins and that they respond to pings at their announced IP addresses. Despite some valid criticisms about its initial distribution, the Dash network has a proven track record of security, and the code has demonstrated that it does the essential functions required.

One element that has been added to this core C++ layer is a new Masternode voting system. In this system, any Masternode can submit a ticket

requesting payment from the network in return for some service, or to be reimbursed for valid expenses incurred in the creation of new functionality. This ticket includes a text field which could contain, for example, a link to a PDF file describing the purpose of the request; for instance, the PDF might detail the listing cost with an exchange, and the submitter of the ticket would try to convince a majority of the other Masternodes to vote in favor of the proposal. In addition, the ticket will include the Pastel blockchain address at which the submitter wants to receive the funds if the ticket is approved.

A total of 5% of the Pastel mining block reward will go towards payments to such addresses (i.e., to the Pastel *Growth Fund*, assuming that they win enough votes using the voting system. In order for a ticket to be approved, it must be voted on by a sufficiently higher number of MNs so that a *quorum* is available– a minimum number below which a vote is invalid. If the quorum is met, then elections are based on a majority vote. Submitted requests arrive in a queue, and are entered into circulation among MNs as part of a list of potential recipients of funds, so that each MN can vote on whether they approve or disapprove of the submitted request. Assuming a ticket is approved, the payments will will be sent in the form of *coinbase* transactions, similar to how mining rewards and Masternode rewards are paid out by the network. If the requested amount exceeds the amount of the next 5% payment, then the ticket continues to remain outstanding until it is fully paid the originally specified number of coins.

8.3 Novel Python Code Base

As described previously, the higher-order functions of the Pastel network– namely, those which operate by means of ticket files written to the blockchain (e.g., artwork registration tickets, artwork trading tickets, identity establishing tickets, etc.)– essentially "piggy back" on the core blockchain functionality of Bitcoin/Z-Cash by using these transactions as "dumb storage."

Just because a ticket file has been written to the blockchain by means of these special coin transactions, it is not necessarily deemed to be valid by Masternodes: first, the ticket file must be carefully and independently reconstructed, parsed, and validated by each Masternode. Once validated, the Masternodes must act on the basis of these tickets, performing a variety of specific services in coordination with other randomly selected MNs.

All of the computer code to facilitate these operations are implemented in the custom *Python* system developed for the Pastel project. The code is written in a modular way, with various utility and other classes to reduce code redundancy. It is also fully open-source, with the updated contents available

to anyone around the world through the *GitHub* code repository service.

9 Pastel ID Digital Signature System

9.1 Overview and Goals of System

The starting point and foundation for the Pastel system is a secure means of identifying network participants that has the following properties:

1. It should be **persistent**– that is, artists, collectors, and Masternode operators should have a single identity that follows them across all of their interactions with the network. This is essential for having a functional reputation tracking system and for detecting and mitigating hostile behavior by malicious users.

    - In the Bitcoin network, there is no overarching concept of identity; although users can sometimes be tracked via IP address, it is easy to take precautions against this, such as the use of virtual private networks (*VPNs*), that would make such tracking impossible.
    - Each Bitcoin address is, in a sense, an "identity" on the network: the owner of such an address– that is, someone who has knowledge of the private key associated with that address, if indeed such an owner exists– can prove ownership, either by actually sending the coins to a specified new address, or by signing a specified message with the corresponding private key to produce a signature that can be verified by third party nodes by comparing the signature against the known public key of the signer.
    - The way this is actually implemented in Bitcoin, as well as in Z-Cash and Dash, is through the use of the *Elliptic Curve Digital Signature Algorithm* (ECDSA). In particular, Bitcoin and others make use of the Secp256k1 curve parameters, which were selected by Satoshi for their lack of connections to so-called "three letter agencies," as well as their computational efficiency compared to other parameters.

2. The ID system should be **decentralized**, with no participant in a privileged position to act as the ultimate authority for the validity of a given ID. In practice, this means that we must use some form of asymmetric or *public key cryptography* to establish an identity and to allow for the generation and verification of secure digital signatures.

3. As far as Masternodes are concerned, any interaction between a valid Masternode and another user– either another MN or an artist/collector– must be signed with a valid Pastel ID in addition to the built-in digital signature scheme, through which a Masternode can prove the ownership of sufficient coins to qualify as a valid MN. That is, for network interactions arising from Masternodes, **two layers** of signatures are always applied to every transaction involving a blockchain "ticket" as detailed previously.

4. The ID system should be built exclusively using well-known, *peer-reviewed cryptographic primitives* (e.g., hash functions, public-private key encryption scheme, etc.), particularly ones that have publicly available reference software implementations that we can use in order to reduce implementation risk.

5. The cryptographic primitives used in the ID system sit on top of similar primitives that are used in the base blockchain layer (i.e., the part built directly using the Bitcoin/Z-Cash architecture). The primitives used in Bitcoin– *SHA2-256* as a hash function and *Secp256k1* as the elliptic curve parameters– are now nearly 10 years old. Since then, the state of the art in cryptography has advanced significantly. Therefore, the new primitives used for the ID system should be **different and complementary** to those used in the underlying blockchain implementation.

9.2 EdDSA and E-521 Meets these Requirements

We believe that the Pastel ID system, described in detail below, satisfies all of these criteria. Essentially, the method we adopt is the peer-reviewed *Edwards-curve Digital Signature Algorithm*, or EdDSA scheme. This system, the precise specifications for which are given in the *RFC 8032* proposal, was publicly submitted by its authors in January of 2017. The EdDSA scheme is similar, and can be considered as an evolutionary improvement over, the older *ECDSA* digital signature scheme used in Bitcoin.

EdDSA was developed by a highly respected team led by Daniel J. Bernstein, considered by many to be the premiere living cryptographer, and an authoritative source on various matters of information security and encryption. The primary improvement of the newer EdDSA system is that it can be computed even more efficiently (without sacrificing security) by taking advantage of certain mathematical symmetries in the underlying elliptic curve geometry. This in turn allows for the use of significantly longer private keys,

which makes the job of cracking the encryption scheme (i.e., comprising the system so that a user's digital signature can be forged by an attacker, without that attacker ever gaining access to the user's private key) much harder, by requiring an exponentially higher amount of computing resources to mount a brute-force attack.

The EdDSA scheme is, by itself, an incomplete specification: one must also select a *hash function* and a set of *curve parameters* to fully determine the system:

1. **Hash Function:** This is used to create a "digest" of a given piece of data (such as a blockchain "ticket") which can then be signed by the user with their private key. The purpose of the hash is to allow users to sign this shorter hash output rather than signing the entire original message. Since a secure hash function should be impossible (or at least practically impossible) to compute in the opposite direction– that is, to find particular input data that results in a specified hash– this allows the hash output to effectively "stand in" for the original content in a secure way.

    - For hash functions, Pastel has selected a combination of the *SHA3-512* hashing algorithm and the *BLAKE2b* hashing algorithm.
    - By separately computing each hash, and joining the resulting strings together, we get a combined hash that combines elements of the security of both systems (i.e., both hashing algorithms would need to be compromised in some way, such as a practical *collision attack*, in order for an attacker to undermine the system; since we only use the first few bytes of the BLAKE2b hash value, this overstates the security benefit in practice.)
    - Both algorithms were selected by standards-making bodies on the basis of well-publicized and public selection processes, with top academic researchers and industry participants involved in the vetting procedures. In addition, both have been used in real-world applications for multiple years (for example, BLAKE2b is used as the proof-of-work algorithm for the SiaCoin and *Decred* crypto-currency projects), further reducing their risk.
    - Both algorithms are also implemented in the built-in Python Standard Library, which is an indication of their degree of maturity and solid reputation by researchers and practitioners.

2. **Curve Parameters:** This specifies the particular parameters (i.e., numbers) used to construct and initialize the specific elliptic curve that the system is based on. Curve parameter selection is a controversial subject, with many expressions of skepticism by leading researchers (notably Dan Bernstein), who claim that the US Government (and specifically the NSA) have used their superior knowledge, techniques, and resources to systematically undermine the curve parameter selection process– even going to great lengths to make the proposed parameters appear on the surface to consist of "nothing up my sleeve numbers," when in reality the parameters were the end result of a massive search across the space of parameters, selected specifically because they were known to be compromised by some proprietary attack known only to the NSA and not to the academic cryptographic research community.

    - In light of the potential risks, we believe that the best strategy is to choose curve parameters that have been tested and certified by Dan Bernstein, as documented in his well-known website, SafeCurves.
    - Ideally, we believe these parameters should employ as large a *private key* as possible. If one goes through all of the curves presented on the *SafeCurves* website as of late 2018, the parameter set featuring the largest key size is presented under the name *E-521*. This name denotes the large prime number used in the algorithm: 2 to the 521st power, minus 1. The fact that such a large prime number could be constructed in such a simple way is a very good indicator that the curve parameters have not been concocted in some special way so as to undermine the security of the system. Furthermore, this prime is much, much larger than the prime used in the curve parameters for Bitcoin's digital signatures, *Secp256k1*: the prime for E-521 is 157 digits long compared to 78 digits long for Secp256k1.
    - E-521, in addition to being featured on the SafeCurves website (as illustrated in the following figure, in which E-521 is listed in the bottom row with the maximum possible number of "green check marks" used by the site to illustrate the security characteristics of each set of parameters), is described in detail in the 2015 paper entitled EdDSA for more curves by Bernstein, et al.

Now that we have specified the hash function and the curve parameters, we are finally talking about a concrete, specific system. The primary purpose of this system, at least as we will be using it, is the production of digital signatures. Digital signatures, in brief, allow a user to demonstrate knowledge of

Figure 6: The *SafeCurves* Project Curve Parameter Summary Table (Source: *safecurves.cr.yp.to/index.html*)

		Parameters:			ECDLP security:				ECC security:			
Curve	Safe?	field	equation	base	rho	transfer	disc	rigid	ladder	twist	complete	ind
Anomalous	False	True✔	True✔	True✔	True✔	False	False	True✔	False	False	False	False
M-221	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
E-222	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
NIST P-224	False	True✔	True✔	True✔	True✔	True✔	True✔	False	False	False	False	False
Curve1174	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
Curve25519	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
BN(2,254)	False	True✔	True✔	True✔	True✔	False	False	True✔	False	False	False	False
brainpoolP256t1	False	True✔	True✔	True✔	True✔	True✔	True✔	True✔	False	False	False	False
ANSSI FRP256v1	False	True✔	True✔	True✔	True✔	True✔	True✔	False	False	False	False	False
NIST P-256	False	True✔	True✔	True✔	True✔	True✔	True✔	False	False	True✔	False	False
secp256k1	False	True✔	True✔	True✔	True✔	True✔	False	True✔	False	True✔	False	False
E-382	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
M-383	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
Curve383187	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
brainpoolP384t1	False	True✔	True✔	True✔	True✔	True✔	True✔	True✔	False	True✔	False	False
NIST P-384	False	True✔	True✔	True✔	True✔	True✔	True✔	False	False	True✔	False	False
Curve41417	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
Ed448-Goldilocks	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
M-511	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔
E-521	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔	True✔

a private key without actually disclosing the key; the user accomplishes this by using the private key to sign a given piece of data– similar to how a person might sign their name to a legal agreement. The signature represents a commitment that cannot later be repudiated, and which could only be produced by someone with knowledge or control over the corresponding private key in the public-private key pair.

The matching functionality, of course, is that a user must also be able to look up a purported signer's public key (in our system, this information is stored in the blockchain itself in the form of an identity establishing ticket, but also potentially available on an artist's external website or social media accounts), and compare that public key to the signed content and the signature string. This allows the user to determine objectively– using purely mathematical operations that are believed by researchers to be impossible to perform in a reasonably amount of time– whether or not the signature is valid. The beauty of the system is that the signer never has to reveal their private key

in this procedure– the signature is enough to verify that the signer controls the private key, and yet does not compromise or reveal information about the private key to a potential attacker. This makes the system trustless, which is a pre-requisite for truly decentralized system.

9.3 A Trusted Implementation of the System

As mentioned previously, a large risk in any cryptographic software system development is the introduction of *implementation risk*; essentially, this means software bugs that can compromise the effective security level of what would otherwise be a theoretically secure system. Such implementation bugs can occur in many forms, ranging from *buffer overflow* and other memory corruption errors to so-called *side-channel* attacks that reveal aspects of the underlying cryptographic computations as a result of "leaking" information in various ways (e.g., revealing the precise amount of time required for each step of the computation, which can be cleverly exploited by a sophisticated attacker to deduce secret information such as the private key).

As a result, it is infinitely preferable to rely on a "known good" implementation: one that has been extensively scrutinized by a large number of qualified researchers and subject to a formal review process through which potential security vulnerabilities could be identified by many independent research teams. To clarify, we are talking here about specific, usable code, not just the theoretical description of the code: transforming such a theoretical specification to a usable implementation is precisely the thing that is so difficult to do correctly, and so fraught with the potential for catastrophic error when done by engineers who are unsophisticated or inexperienced in theoretical cryptography (cautionary tales of this kind abound; for example, the notorious security lapses discovered in the *CryptoCat* secret chat software).

Fortunately, the above mentioned RFC (i.e., RFC-8032), in addition to containing a complete theoretical specification of the system, also includes an appendix that provides an extremely clear and concise implementation of the system in the Python language. When only the parameters and hash functions used by the Pastel ID system are included, the entire commented code base for this functionality weighs in at under 400 lines of code, and requires only the Python Standard Library as a dependency. Again, this code is taken directly from the RFC without modification, which the reader can readily verify by inspecting the Pastel source-code repository and comparing it to the code from the RFC document. To sum up, we are able to use a relatively small, self-contained code base that has had a lot of "eyes" on it for some time and found to be correct and secure (or at least, it has not yet proven to be

unsecure!).

In addition, we take further measures in our use of this code in the Pastel system. For example, in order to reduce the risk of side-channel attacks, we wait for a short, random period of time before and after performing any cryptographically sensitive computation (this risk of side-channel attacks is generally why statically-compiled versions of cryptographic algorithms, written in languages such as C++, are usually preferable to those written in dynamic language such as Python).

In order to effectively maintain the secrecy of the Pastel ID private key, we use the method described earlier, in which the user can store his private key on a smartphone, in the form of a special video file that displays an animated QR code which contains the private key. The user can thus convey the private key from their phone to their computer using only the computer's built-in web camera, effectively "air-gapping" the communication of the key, so that no direct network/electronic access is required. This obviates the need for typing a secret password, which might be intercepted by malware running on the client machine without the user's knowledge. Another benefit of this approach is that *no record of the private key is every stored on the hard disk of the computer*, so that when the wallet software is closed, the key must be re-entered before the user can be authenticated.

9.4 The Usefulness of an ID/Signature System

Now that we have a secure, solid foundation for an identity system, we can use it in various parts of the system's architecture. In particular, we can use the Pastel ID to ensure the validity and to prove the authorship of tickets written to the network's blockchain based system, as well as to authenticate the plaint-text "messages" described in detail in the next section, which allow nodes to securely communicate with one another without making use of the underlying blockchain.

The Pastel ID is also what makes possible the implementation of a decentralized reputation tracking system, which is a powerful tool for constructing the incentive structures required to drive useful and honest (i.e., non-malicious) behavior by network participants. By creating a series of "carrots" and "sticks" (i.e., rewards and punishments), such a reputation tracking system can make undesirable actions, such as Masternode that are slow to respond to ping requests or storage challenges by other MNs, expensive for the owners of such Masternodes. At the same time, it can make desirable behavior– such as exceptionally fast and accurate network interactions– revenue *enhancing* for honest Masternode operators.

9.5 The Pastel ID as the Basis of a Reputation Tracking System

By attaching the reputation tracking system to the persistent Pastel ID, rather than basing it entirely on the built-in MN collateral address signature system, we mitigate the risk of a so-called *Whitewashing Attack*, in which a malicious MN owner exhibits bad behavior that result in the deterioration of that MN's reputation, but can then avoid the repercussions of those bad actions. The malicious MN owner does this by effectively "laundering" their coins by sending the associated MN collateral to a crypto-currency exchange site, where it can be commingled with coins from other users. If the malicious MN owner were to then withdraw the same number of coins to a newly created blockchain address, there would be no way for other network participants to know that the bad actor is the same entity as before. For more details on this attack vector, the reader can consult the 2015 paper entitled Defending Against Whitewashing Attacks in Peer-to-Peer File-Sharing Networks.

Such a reputation tracking allows for the introduction of "trusted" Masternodes– that is, Masternodes that every node can independently verify– through the careful monitoring and analysis of log files that describe the behavior of each node, gathered automatically in the background during the normal operation of the Masternode software– have been acting in accordance with the expected norms of an honest MN. By this we mean, for example, that the Masternode in question has historically responded quickly to ping requests, that it has successfully passed off-chain storage challenge requests, and that it has not otherwise engaged in harmful or dishonest/questionable behavior on the network. As more and more time goes by with a Masternode maintaining and enhancing their reputation, their *reputation score* would increase.

So why should a MN owner care about their reputation on the network? One good way to make the MN owner care is to restrict certain activities so that they are only available to Masternode with a high enough reputation score. For example, the opportunity for a Masternode to be part of the pool of potential "winners" of the right to act as a broker in the exchange of a rare artwork (and to receive the associated brokerage or trading fees for providing this service) could be apportioned only to trusted MNs. This would enhance the income potential of such MNs compared to a newly-registered MN that had a lower reputation score, creating a strong incentive for MN owners to act in a way that is in the interest of the network as a whole.

By establishing such an incentive system, we make it profitable to be honest. More importantly, we make it so that the Pastel ID becomes valuable apart from the value of the Masternode stake itself. Thus a rational Mas-

ternode operator should be reluctant to "start over" in the network as a new MN with no reputation; this offsets the potential profit available to malicious nodes that might otherwise be tempted to engage in whitewashing attacks. Put differently, why would a rational MN operator ever choose to walk away from a good reputation, which is effectively what they would be doing by engaging in dishonest or destructive behavior, when they can make even more money from the system by being honest? In this way, we effectively raise the bar required for the potential payoff of any dishonest act, by offsetting it with a corresponding loss in value from the reduced income resulting from a decreased reputation score. And because all Masternode-related messages are signed with the MN collateral address key in addition to the Pastel ID private key, our proposed system offers all the same security guarantees as the core C++ Masternode authorization and management code largely taken from the Dash project, while augmenting this with an additional layer of security.

10 Masternode Messaging System

10.1 The Need for Messaging

Ultimately, the "ground source of truth" in a blockchain project should always be the blockchain itself. But as mentioned earlier, space in the blockchain, insofar as storing arbitrary data is concerned, is an extraordinarily precious and limited resource that we must use as sparingly as possible. And yet, there are various aspects involved in regulating the network that would be much easier to implement if we had a reliable way for network participants to communicate with each other. But the storage requirements for such messages would really add up over time if we tried to use the blockchain for all of these messages. Furthermore, many of the messages would be ephemeral– that is, useful to the network only for a brief period– thus it would be a waste to store these permanently as bits on the blockchain. All of this suggests that it would be useful to have a separate, parallel system through which MNs and users could communicate in an off-chain context. If we consider what properties would be desirable in such a system, the following come to mind:

1. It should be as simple as possible, so as to minimize the attack surface. Messages should be completely passive (i.e., it should be impossible for executable binary code to be contained in a message) and should also carry a relatively short "message payload" to further mitigate DDOS attack risks and other potential security exploits.

2. It should be constructed using well-known libraries that are relied upon by several high-profile open-source projects in order to reduce implementation risk. These components should also be performance-oriented, so that the resulting messaging system does not create a scalability bottleneck in the network.

3. It should essentially resemble other parts of the project's architecture wherever practicable, so that we can keep the number of separate systems required to a minimal level.

4. Finally, observe that *we do not require encrypted communications*. Indeed, it is better for the overall system security if messages are publicly readable by any network participant. In particular, the unencrypted message data can drive the reputation tracking system, since this data contains very relevant information: it shows which nodes requested which services, and also the timing and content of a node's responses to incoming messages. What we *really* care about here is that we can verify the authorship and integrity of messages: that is, that a given message was actually written by the purported author, and that the message was not modified in any way in transit.

10.2 Technical Architecture

These considerations drive the main design decisions used in the novel Pastel messaging system:

- We use the **MessagePack** serialization library for encapsulating the messages. According to the library's authors, "MessagePack is an efficient binary serialization format. It lets you exchange data among multiple languages like JSON. But it's faster and smaller." The MessagePack project features interfaces for all modern computer languages, and is used in such high-profile open-source projects as *Redis* (a popular in-memory database) and *Fluentd* (a data collection and logging system). The use of a trusted library is critical from a security standpoint, as serialization and parsing are inherently dangerous activities from a system security standpoint, particularly when inputs come from potentially malicious users. A parsing bug or buffer-overflow/remote-execution bug could prove catastrophic, leading to loss of user funds in the worst case. Furthermore, serialization/parsing is notorious in security circles for being a weak point; in the Python language, for example, the *Pickle*

serialization library has historically accounted for a wildly disproportionate share in critical security vulnerabilities found in the Python standard library.

- We use the **ZeroMQ** library for handling the network communications between nodes. This library is highly respected in the software development community, and is even used in the Bitcoin code base to a limited degree. The primary advantage of using ZeroMQ in the Pastel project is that it is fast, minimalistic, reliable, and flexible. In particular, ZeroMQ supports various common patterns that arise in typical communication scenarios, such as direct node-to-node messaging, publisher-subscriber ("pub-sub") messaging, "multi-cast" messages to the entire network, and so on. According to ZeroMQ's authors, the primary selling points of ZeroMQ are the following:
  - Connect your code in any language, on any platform.
  - Carries messages across inproc, IPC, TCP, TIPC, multicast.
  - Smart patterns like pub-sub, push-pull, and router-dealer.
  - High-speed asynchronous I/O engines, in a tiny library.
  - Backed by a large and active open source community.
  - Supports every modern language and platform.
  - Build any architecture: centralized, distributed, small, or large.
- While Pastel messages are transmitted in clear-text (i.e., in unencrypted form), all messages are signed using the Pastel ID system to ensure the identity of senders. In addition, any messages sent by a valid Masternode would also include a signature based on the private key corresponding to that Masternode's collateral address; this signature proves that the sender has control over the required coin balance.

So what are these messages used for in the system? Although the messaging system is intentionally designed in a extensible way, the initial functionality that will be implemented via the messaging system includes:

- **Any sort of coordination that needs to be done by network participants for the purposes of preparing a blockchain ticket**. For example, before an art registration ticket can be finalized on the blockchain, it must be signed by each of the various selected MNs. That is, we can use messages to "pass around" the ticket for everyone to sign.

- **Any request for services on the network**. For example, a request that a Masternode send a particular chunk file in the off-chain storage system, or a request that a trade ticket be created to facilitate the exchange of an artwork. Similarly, any advertisement of services or other content to the network (such as the dissemination by a Masternode of a new trade request when that MN is acting as a broker on behalf of an artist or collector) would be conducted via the messaging system.

- **All forms of challenge-response communication between MNs**. Essentially, any service offered by a Masternode, such as duplicate image detection or the storage of file chunks in the off-chain storage system, must have a corresponding challenge mechanism in place to ensure that nodes are actually performing the required services. These challenges should be as efficient as possible for scalability purposes. For example, to verify that a given MN is actually hosting a specific file chunk in the off-chain storage system, we could simply ask the MN to send the entire file– but this would be very wasteful of bandwidth. Instead, we can request the SHA-256 hash of a particular random segment of the chunk file, a request that can be efficiently communicated and which is quick and easy for the challenged MN to respond to if and only if the MN actually does have the file at hand. This allows us to establish that the node is acting "honestly," at least with regards to that specific file chunk.

While each of the above processes is essential to the proper operation of the network, they do not necessarily require all of the same security guarantees that we get from a block-chain transaction. Because all messages are signed using, at a minimum, a user's Pastel ID, messages have a "base-line level" of security that allows us to rely on them for regulating behavior between nodes on the network.

Ultimately, what really matters is always the content of the tickets written to the actual blockchain via valid tickets. Where the messaging system comes into play is in reaching consensus across nodes of what that content should consist of, as well as parsing and validating this content. The fact that ZeroMQ allows for flexible network topologies means that, for each kind of communication modality we want to use– pub-sub, push-pull, multi-cast, etc.– we can do it efficiently and with high performance and reliability without concerning ourselves with the complex implementation details of how to do this at a low level, such as retrying lost packets. And by sending messages in plain-text, we effectively side-step the limited cryptographic options in ZeroMQ by bringing

our own digital signatures into the system at a higher level of abstraction (i.e., in the message payloads themselves, rather than using a "wire protocol" built into the networking code.

11 The Blockchain Ticketing System

11.1 Overview

Perhaps the single most critical principle in the entire architecture of the Pastel project is that we want to treat the underlying blockchain as a kind of "read only" cloud storage drive, and then build all the higher-order objects in our system, such as the underlying representation of rare artworks, at a more abstract level– in particular, in the form of a passive ticket, which is essentially a simple table structure containing fields and their corresponding values.

That is, we do not attempt to stuff all of the various aspects of a particular artwork, such as its title, the image file hash, the quantity of digital prints to create, etc., into an actual Bitcoin-like transaction. This "stuff everything in one transaction" approach is employed by the *CounterParty* project, and invariably leads to registered assets suffering from a variety of limitations. For example, assets registered using the CounterParty protocol may only use names consisting of fewer than 12 upper-case Latin letters, limiting its utility dramatically, especially among foreign users who do not speak English.

Instead, we take a very different approach: we treat the transactions as "dumb bits." By this we mean that the transactions we use merely treat the coins as a vessel for encapsulating unrelated, arbitrary data. This frees us from all the limitations of fitting any meaningful arbitrary data within the confines of a Bitcoin-style transaction, which means that, among other things, artworks registered in this manner can feature much longer titles and can include any *Unicode* characters, making the system more practical for international use across a variety of languages. Indeed, our approach means that we can fit in a variety of optional fields, such as a brief written statement from the artist.

As mentioned previously, Bitcoin was design as perfect e-Gold, and it excels in that function. As a result of this focus, Bitcoin's transaction format is stripped down, with the binary representation as efficient as possible so that no space is wasted: essentially, only the minimal number of fields required to keep track of particular amounts of coins are included in the transactions. Although Bitcoin does have a scripting system as part of the transaction specification, it is critically limited and not Turing-complete, which makes it much easier to reason about from a security standpoint that *Ethereum* and other "smart

contract" systems.

Though there is an admirable desire for simplicity displayed in CounterParty's approach to asset creation and trading directly "inside" the environs of Bitcoin's core architecture (i.e., the CounterParty protocol exists at the same level of abstraction as the raw blockchain transactions themselves), it simply is not workable in practice: space is too limited. This shouldn't be surprising, since that is not what the system was designed to do. Although we could attempt to circumvent this by altering the underlying C++ blockchain code (something that could not be done by CounterParty, since that system is actually built on top of Bitcoin, which the CounterParty developers could not change), but doing so would be the equivalent of performing brain surgery on a patient, leading to a morass of potential security and integrity concerns. It would also make the introduction of future improvements and changes in the Bitcoin and Z-Cash code bases much harder to integrate into the Pastel project by "breaking" core aspects of the architecture, and would also render ancillary tools such as block explorers and mining pool software incompatible.

In our view, it is better to not "mess with the winning formula" of Bitcoin and Z-Cash. By keeping all the core blockchain code the same as in these well-tested and proven platforms, we position our system on "terra firma" that we can reasonably rely on to support the more complex structures that we want to build. By instead treating Bitcoin-like transactions as simple building blocks– that is, as arbitrary bytes on a disk that anyone can later read from in order to reconstruct that same pattern of bytes– we inherit the underlying immutability and security of the blockchain code. But of course, this does not imply that the resulting system will, as a whole, be secure: we must also be careful in how we design and implement these higher-order abstractions, since these introduce additional attack vectors. Although such vectors may not have any bearing on the integrity of simple value transfers on the blockchain– i.e., on the kinds of operations regulated by the underlying blockchain code– they could undermine the security of the abstract objects that we construct using these lower-level blockchain transactions, such as the artwork registration or trade tickets.

11.2 Core Architecture of Blockchain Storage

So how does all of this work in practice? Although there are a variety of published methods for writing arbitrary data to a Bitcoin-like blockchain, they each operate on the basis of the same underlying principles:

1. First, we take the raw data that we want to store, and express it using an encoding format that makes use of the same characters that are allowed

in ordinary blockchain addresses; roughly speaking, these include upper and lower-case letters and numbers. The result of this step is a text string containing a representation of the data we want to write to the blockchain.

2. We must now try to break up this string into a series of segments, and then attempt to make each segment appear to be a valid blockchain address. Valid addresses must satisfy certain conditions: for example, Bitcoin addresses must begin with a "1" or "3," and certain potentially confusing characters, such as lower-case "L" and upper-case "O" are not permitted. Thus we do not have complete freedom over every part of a generated address.

    - Despite these constraints, it is relatively straightforward to come up with a scheme in which the data can be split into strings of characters which can then be "hidden inside" a valid looking address, creating what we can call a "fake address."
    - Importantly, such blockchain addresses are fundamentally different from those used in everyday usage for transactions: while addresses are usually generated in connection with a particular private key, so that the creator of the address can maintain control over any coins sent to the newly generated address, this does not work in reverse: if one first specifies the address, then it is impossible to know the corresponding private key associated with that address.
    - Indeed, even specifying the first few characters of a blockchain address, as is commonly done in Bitcoin for the generation of so-called "vanity addresses," can require an astronomical amount of time if we want to specify more than a handful of characters, since this requires a massive brute force exploration of the space of possible keys.
    - As a result, any coins sent to a "fake address" in this manner are necessarily *unspendable*; while the coins continue to exist on the blockchain, they are not accessible to anyone because no one knows the private keys that would unlock them.

3. The reason for this previous step is that these "fake addresses" are how we actually store the data: by sending a tiny, economically insignificant amount of coin to a large number of fake addresses, we can cause our arbitrary data to be encoded in the blockchain. The resulting transaction

ID (TXID) that arises from sending this small amount of coin to all these addresses can then serve as a unique filename or "URL" for the data. That is, any node could take this TXID and look it up using the built-in RPC system in the wallet software or on an external block explorer website and get back the list of addresses.

4. By taking this list of addresses and processing and "reversing" the steps taken in the original encoding process, any node can reliably reconstruct the arbitrary data written to the blockchain. Effectively, this turns the blockchain into a kind of hard drive that can only be written to a single time. The data written in this manner can never be modified or deleted, any more than a valid Bitcoin transaction can be reversed after the fact. Since the coins used in this data recording process continue to exist in network, they continue to be tracked by all nodes "forever."

As discussed above, there are a variety of methodologies for accomplishing these steps. The following figure, taken from the most comprehensive work published on this subject (the 2017 paper entitled Data Insertion in Bitcoin's Blockchain by Sward et al.) lists all of the known methods and illustrates the amount of data that can be stored using each method (and also the *storage efficiency* achieved):

While a complete discussion of these methods is beyond the scope of this discussion, they each take advantage of different aspects of Bitcoin transactions to store data. The primary difference between the methods is in which particular transaction fields they use for storing the arbitrary data. Another critical difference between the methods is whether or not nodes are made aware that the coins involved in a such transaction are unspendable. This functionality, known as *OP_RETURN*, was introduced in the Bitcoin software in response to users writing data to the blockchain in the manner discussed above. This usage was viewed as being undesirable by the Bitcoin development team for scalability reasons. By intentionally marking such transactions as using unspendable coins, OP_RETURN allows these transactions to be "pruned" by nodes in order to reduce the amount of data stored in system memory.

Given the primacy of the artwork related information in the Pastel system, we concluded that the system should store this data in the most fool-proof way: in the form of the set of "unspent transaction outputs" or *UTXO Set*. Such transaction data is stored directly in the high-performance, memory-backed database used in the Bitcoin software implementation (i.e., *LevelDB*), which means that any full node will always have ready access this data at a moment's notice. Of all the methods described in the figure, the one that achieves the

Figure 7: A Comparison of Blockchain Storage Methodologies (Source: *DOI 10.5915/LEDGER.2018.101*)

Table 3. Method Summary (Max Size and Cost) for a Fee of 20 Satoshi/byte

Method	Stored in UTXO Set?	Max Data Per TX*	Total Cost **	Data Eff.	Cost Eff.***
PF2KH	Yes	58,680	.03601624	58.7%	61.30
PF2K	Yes	85,280	.02715132	85.3%	31.80
OP_RETURN	Prunable	80	.00006340	25.2%	79.25
P2FMS	Yes	92,624	.02522220	92.6%	27.23
P2FSH	Yes	62,340	.03701302	62.3%	59.37
Data drop (w/o Sig)	No	96,060	.02042260	94.1%	21.26
Data Drop (w/ Sig)	No	90,099	.02042260	88.2%	22.66
Data Hash (w/o Sig)	No	92,507	.02042900	90.5%	22.08
Data Hash (w/ Sig)	No	86,087	.02042260	84.3%	23.72

* Data in Bytes **Cost in Bitcoin

*** Efficiency in Satoshi per Byte of arbitrary data stored

ISSN 2379-5980 (online)
DOI 10.5915/LEDGER.2018.101

highest level of storage efficiency while using the UTXO Set for storage, is known as *Pay-to-Fake-MultiSig*, or **P2FMS**.

Why is this method the most efficient? Multi-signature addresses, which were added to later versions of the Bitcoin software, allow a user to send coins to an address which requires multiple private keys, each of which are generally known to a single independent entity. For example, a non-profit foundation that accepts donations in Bitcoin might not want all such donated funds to be exclusively under the control of a single person, which might result in embezzlement. Instead, the foundation could use a "3-of-5" multi-sig wallet setup, so that 5 of its board members could serve as key-holders, so that funds can only be disbursed from the address if at least 3 of the keys can be assembled (i.e., if 3 of the board members authorize the transaction). This is analogous to the secure systems used for launching nuclear missiles, where two independent technicians must simultaneously insert special keys in order to authorize a launch.

So why is this relevant to us? Well, multi-sig transactions allow one to specify more than one address at a time, and since we are literally storing our

arbitrary data using the address field, cramming more of these addresses in the same envelope means that we are wasting less space on "protocol overhead." That is, more of the data actually stored in the blockchain is comprised of the data we care about, rather than wasted on ancillary fields that are merely a by-product of storing the data in this manner. In any case, P2FMS allows us to achieve a maximum ticket size of 92.6 kilobytes at an efficiency rate of 93%, which is surprisingly high. It is also by far the most cost-effective of the methods that store data in the UTXO set in terms of the amount of coin that must be burned in order to write a given amount of data. As such, we feel that the P2FMS approach is the optimal methodology to use.

11.3 The Representation of Tickets on the Blockchain

Of course, Pay-to-Fake-MultiSig simply offers us a means of writing data of any kind to the blockchain. In order to use this for constructing rare digital artworks, we must first choose an appropriate representation for the registration tickets. Many of the considerations that guided us in selecting the format used in the off-chain messaging system apply equally well to the tickets stored in the blockchain. That is, we want them to be simple and secure, and to be built using industry-standard libraries. Thus, we mostly employ the same tools that we used for representing messages: we express blockchain tickets using the *MessagePack* serialization library.

In brief, this gives us a very stripped down and simple implementation that is similar to the familiar JSON format employed universally in the web development community: we can specify fields or attributes (e.g., the name of the artist, the title of the piece, the website of the artist, etc.), and for each of these we can specify a corresponding value for that attribute (e.g., "Pablo Picasso", "Guernica", etc.).

JSON is popular in part because it is so simple and "safe" from a security standpoint, but this safety comes at a cost: since JSON is a text-based format rather than a binary format (which can be designed more efficiently) it is "wasteful" of space, which is at a premium on the blockchain. MessagePack is basically like JSON except that it allows for a more efficient representation of data contained in fields that do not require as much precision. For example, an integer value such as "100" can occupy fewer bits than if the string "100" were stored as text using a typical encoding such as *UTF-8*.

Like messages, blockchain tickets must also be signed by all relevant parties. For example, in order for an art registration ticket to be validated by nodes as a legitimate registered artwork, it must contain the signatures of both the artist and the particular set of MNs selected through the rank ordering method to

register new artworks submitted during that particular block. These signatures are written on the hash of the contents of the various metadata fields of the ticket, so that if any of the information contained in these fields were to change, the resulting hash would change, which would render the digital signatures invalid and require the generation of new signatures by every user involved in the transaction. Since the private keys required to generate such signatures are presumed to be known only to the relevant users, such a system offers a secure way to determine the consensus of the relevant network participants on the precise contents of a given blockchain ticket.

11.4 The Generality of the Ticket Concept

The kind of tickets described above provide a powerful and generic framework for encapsulating all sorts of structure and functionality that we might want to have in our system. That is, we don't need to use these tickets only for art registration: any sort of process that can be rigorously defined across a set of nodes, where the nodes must exchange bits of data with one another, can be implemented by means of simple tickets on the blockchain. The catch is that the tickets only represent part of the solution: we must also construct an appropriate set of *validation procedures* so that nodes can independently verify that such tickets are correct.

Since the validation of tickets is effectively at the discretion of each node, this is a fundamentally different security paradigm than that found in *smart contracts* of the kind used in Ethereum and other "smart-contract" projects. Unlike Ethereum smart-contracts, which are *active* computer programs, Pastel tickets are *passive*, which limits the amount of harm an dishonest node can do to the network by crafting a malicious ticket. It also makes the system much easier to reason about, even by ordinary users. Although the potential for complexity and flexibility can initially appear attractive in a blockchain design, this complexity always comes at a huge price: it becomes impossible for any individual– even a highly competent engineer– to fully comprehend all the inter-connections and complex relationships involved in a dynamic, unconstrained system such as the ones found in any moderately complex, smart-contract driven "DAPP" (i.e., decentralized application).

That's why all sufficiently complex software systems, from operating systems to web browsers, tend to suffer from an assortment of bugs of various degrees of severity. History has proven again and again that eliminating *all* bugs is extremely difficult and expensive, even for a small code-base. Although 100% "bug free" code is attempted in certain contexts, such as the safety-critical avionics software used in planes and spacecraft, it is generally

understood by knowledgeable practitioners that bugs can and do happen, and there is little one can do to insulate oneself short of employing a heroic and expensive process similar to the ones used by *NASA* and *Boeing* to create fault-tolerant, 100% reliable systems.

And yet, in the blockchain context, we generally cannot tolerate any meaningful bugs without potentially exposing the entire system to a host of security threats, the most critical of which can result in the loss of user funds or assets. Thus it behooves us as developers to take a defensive stance in our design, and draw a sharp line between code and data. We do this to protect users from themselves, by making it difficult to screw things up too much: we let the tickets represent our objects, but then control how these objects are interpreted in the client software itself, rather than dynamically using the blockchain to control the execution and evolution of the various computational processes as would be done in a system such as Ethereum.

Not only is the excessive complexity and uncertainty offered by a Turing-complete system like Ethereum undesirable from a security standpoint, this flexibility is also not required in our application of rare digital art. That is to say, we already know what we need to do to accomplish the various steps involved in making a workable and secure decentralized art registration system, such as NSFW detection. So why wouldn't we just express the required ideas using a regular programming language, such as Python, which is infinitely faster and more efficient, rather than a specialized smart-contract language such as Ethereum's *Solidity* language? Indeed, as we will see in later sections, the computations involved in actually running these steps are highly non-trivial, and would be impossibly expensive to execute in Ethereum due to the excessive cost of the "gas" required for each computational instruction used in the execution of a smart contract's code.

But what if we later wanted to change our specification or add new features? In this case, we can simply update the ticket format and the validation procedures used in the software that all the nodes choose to run and release this in a subsequent version of the Pastel software. We can also include ticket version numbers, so that any such future changes to the system are fully "backwards-compatible."

11.5 The Downside of Flexibility and Complexity

So why is our proposed architecture better? Well, what if there is a mistake or bug in our software implementation? Of course, we endeavor to reduce this risk to the greatest degree possible through prudent management and testing, but bugs are a fact of life. How much damage could someone really do if

tickets are purely static data fields? In the case of a Turing-complete system, such bugs can be catastrophic. One of the most extreme cautionary tales in recent memory is that of the *Parity* multi-sig wallet vulnerability, in which a malicious user was able to globally freeze the funds of thousands of users, impacting a staggering $150 million worth of crypto-currency at the time, according to one estimate.

Of particular note is that one of those most seriously harmed by this security breach was none other than *Gavin Wood*, one of the co-creators and primary designers of the Ethereum network, whose *Polkadot* project lost access to tens of millions of dollars of ICO proceeds. That such a knowledgeable and objectively brilliant individual could nevertheless manage to lose so much money for himself and his investors because of a minor bug– a mere careless oversight– strikes us as "prima facie" evidence that any such approach is hopelessly doomed to be insecure. In any case, it is far too dangerous for use by ordinary users: after all, if the experts can't make it work, what hope do regular users have?

The kinds of failure modes present in the proposed Pastel design are far more benign than in the active smart-contract context. That is, while users in our system might find themselves unable to write a particular ticket for some reason or another, or a ticket could be written that, due to an error in preparation, is deemed invalid by the network, under no circumstances could a user "lose money" in the form of frozen or stolen funds (i.e., coins), since this aspect of the system is implemented at the low-level blockchain layer that we know to be secure and reliable. Similarly, tickets written using to the blockchain using P2FMS automatically inherit the "double spending" protection afforded to blockchain transactions, at least assuming users wait for a reasonable number of confirmation blocks. That is to say, Bitcoin's functionality as a decentralized time-stamping server carries over to the contents of tickets, so that we can establish precedence in the system in an unambiguous way and thereby always reach consensus.

11.6 Types of Tickets

We are now ready to get into the details of the various kinds of tickets used in the Pastel system, and how each is specified and validated. The following list gives the set of all tickets currently contemplated in the initial release of the Pastel software:

1. **Identity Establishing Tickets:** These tickets are used to generate and register a new Pastel ID, which consists of both a public and private

key, in the network. This ticket is special in that it is the only class of ticket which may be written by an ordinary user, as opposed to by a Masternode acting on behalf of an ordinary user. The same basic form of ticket is used by both regular users (i.e., artists and collectors) and Masternodes, with the difference that the tickets written by MNs must also be signed using the appropriate collateral address signature for that MN. Such tickets are comprised of the following fields:

- *The sender's blockchain address*; in the case of a Masternode, this would be the collateral address containing the MN funds; for an ordinary user, it would be the address they used to send coins to the fake addresses in order to store the identity ticket data. This would be validated by other nodes by checking that it is a valid blockchain address.
- *The Pastel ID Public Key*; this long number would be generated by the user's wallet software, and can easily be validated to conform to the expected format.
- *The precise time when the ticket was submitted*; this time-stamp can be validated by other nodes by comparing to their estimate of the current time; if these differ by an excessive amount, the ticket is rejected as being invalid.
- *The Digital Signature on the hash of the above fields*; by taking the hash of the data contained in the above fields, and then signing this hash with the private key associated with the Pastel ID public key listed in the ticket, the user can automatically create the signature, which consists of a string of alphanumerical characters. This string can be automatically validated by other users by first confirming that the hash matches the field data, and then by seeing if the signature string checks out when verified by comparing to the provided public key.

2. **Artwork Registration Tickets:** These tickets embody the core of the Pastel architecture, since they are the ones that control the properties of the most fundamental objects in the system: the artworks themselves. By defining the essential characteristics of a given artwork in an space efficient, yet robust and discriminative way, the ticket allows us to strike a good compromise between security and scalability–that is, to store only what we really need to in the blockchain. The various fields that comprise the artwork registration or meta-data ticket are:

- *The Submitter's Pastel ID Public Key (mandatory)*; This can be incorporated by reference to the TXID of the corresponding Identity Establishing Ticket.
- *The precise time when the ticket was submitted (mandatory)*; this is the same as described above in the Identity Establishing Tickets section.
- *The Artist's Name (mandatory)*;
- *The Title of the Artwork (mandatory)*;
- *The Number of Copies of the Artwork to Create (mandatory)*;
- *The Artwork Series Name (optional)* ;
- *The Artist's Website (optional)* ;
- *A Short Written Statement about the Artwork (optional)* ;
- *A URL with a creation video of the artwork (optional)* ;
- *A Set of Key-Words or Tags that Pertain to the Content (optional)*;
- *The SHA3-512 Hash of the Art Image File in Bitmap (BMP) Format (mandatory)*;
- *The Image Fingerprint Vector (mandatory– the Basis of the Duplicate Detection System)*;
- *The Computed NSFW Score of the Art Image File (mandatory)*;
- *A Set of LT Chunk Hashes Corresponding to the Image File (used as File Names/Addresses in the Off-Chain Storage System)*;
- *A Set of LT Chunk "Random Seeds' (used to regenerate lost LT Chunks in the Off-Chain Storage System)*;
- *A Set of Digital Signatures from the Submitter and the Designated Registering Masternodes*;

  In practice, the Art Registration Ticket really consists of two separate tickets; one of them is the confirmation ticket.

3. **Artwork Trade Tickets:** These tickets are used to
4. **Objectionable Content Take-Down Request Tickets:**

11.7 Compression with Z-Standard

As we have highlighted many times, space is at a premium in the blockchain. Thus we have already described several of the steps we have taken to reduce the amount of space occupied by art registration tickets: we offloaded the image files themselves to the off-chain storage system, and we chose an efficient binary representation for tickets using the MessagePack serialization library. But we should try to go even further: anything we can do to wring out a additional boost to space-efficiency should be considered, since over the long term, this could result in large accumulated dividends in the form of increased scalability and runway. In that spirit, we introduce the idea of compressing ticket files before they are written to the blockchain using P2FMS.

Like most things in engineering, lossless compression represents a series of trade-offs or compromises between compression efficiency on the one hand and CPU/memory intensity on the other hand. Generally, to get the most impressive rates of compression, we must use a high amount of computing power, and the intermediate data structures that are involved can require a large amount of a computer's RAM to store.

From the outset, it's important to remember what sorts of things compression can help with, and when compression either does nothing or even hurts us. Compression works best on structured, repetitive data, such as English text, which can often be dramatically reduced in size by finding clever ways to incorporate repeated bits of data by referencing them. Thus it's only natural that compression will tend to work poorly when applied to "random" unstructured data, also known as *high-entropy* data. Examples of high-entropy data include file hashes, public keys, and digital signatures, which appear on the surface to be random. Of course, if a file hash is repeated several times in the data, compression will be able to shrink the data by storing the hash only once. But the general incompressibility of such data creates an upper-bound on how much we can expect to benefit by compressing tickets.

The other impediment to achieving good compression rates is the size of the input data. If we attempt to compress, say, the English language text of the novel *War and Peace*, then this will compress very well. The reason is that there is so much repetition: there are only so many English words, and any proper names (such as people and places) only need to be stored a single time and can then be referenced. On the other hand, a very short piece of input data (say, a few sentences of English text) will *not* present the same opportunities for a compression algorithm; that's because compression, as generally thought about, is done "in a vacuum": the compressed file is treated in isolation when it comes time to decompress it, with no reference made to the outside world.

While this results in a straightforward system, it limits what we can do.

These considerations have led us to choose the open-source *Z-Standard* compression library for compressing Pastel ticket files. Z-Standard has been developed over several years under the auspices of *Facebook* and represents a compelling "next-generation" compression scheme. The most important selling point for Z-Standard (at least in terms of its suitability for our purposes) is that, when used at the highest compression setting of "level 22," it is able to achieve world-class compression rates in real-world scenarios. For example, a Linux distribution ISO file can be compressed to a ratio of over 12 to 1 using Z-Standard, while more familiar compression algorithms such as ZIP and RAR achieve under 8.5 to 1 compression ratios. This compression power comes at a cost, of course, in the form of sharply increased CPU and memory needs for the compression process (the decompression process, by comparison, is not nearly as CPU intensive). However, when we consider that the typical size of a blockchain ticket is minuscule, even these high requirements are easily manageable with modern hardware.

The other feature of Z-Standard that makes it particularly compelling for the kinds of short ticket files we will be compressing is its support for *compression dictionaries*. The idea of a compression dictionary is simple: rather than treat compressed files in isolation when we want to decompress them, we instead bundle the compressed file together with another, static file, called the compression dictionary. The dictionary file includes whatever data we want to put in it. To the extent that there is any redundancy between the dictionary file and the ticket file we want to compress, the dictionary file allows us to make the compressed file even smaller than it would be when compressed in isolation since we can reference data from the dictionary file instead of storing it all in the compressed file.

So what sorts of data should we put in the dictionary? To start with, we could take a collection of sample ticket files and put these in the dictionary. This would save a lot of space, since any repetitive fields in the tickets, such as "Artist's Name" or "Title of Artwork," could be stored a single time in the dictionary file rather than wasting space in every single ticket that we write to the blockchain. In addition, it makes sense to add to the dictionary file various lists of words and names in common languages such as English and Japanese. That way, if any of these terms are later used by artists in the titles or other fields of the artwork metadata, we can incorporate them by reference instead of storing them again and again. Our experiments show that the use of a compression dictionary, which would always be distributed as part of the software system (with the hash of the dictionary file encoded in the blockchain for security verification), can save up to 30% in terms of increased

storage capacity (CHECK THIS).

Our plan is to freeze the dictionary file in the version 1.0 release of the Pastel software, and then in later releases, update the dictionary file by incorporating the incremental data that has been written to the blockchain between release versions. This approach will allow for the system to become more and more space-efficient over time. To the extent that the same users– artists, collectors, and Masternodes– repeatedly interact with the system, their future interactions will take up less space, since we won't need to keep repeatedly storing their public keys and blockchain addresses. Such changes to the compression dictionary will be made sparingly to keep things as simple as possible, and all the "old" dictionary files will continue to be included in the software distributions so that we can always preserve backwards compatibility with older tickets that were compressed using a previous version of the dictionary file.

11.8 Local Database Reconstruction of Network Ticket History

Thus far, we have described how data is written to the blockchain as well as the format of the ticket data that we are writing. While our proposed architecture works and is secure, it is a very unwieldy representation of the underlying data that we care about. It serves its purpose well in allowing any node to reliably parse, reconstruct, and validate the ticket file and all its fields, but once a node has done so, it doesn't make a lot of sense to keep the data in the native format in which it is stored in the blockchain, since this format makes it inconvenient to do computations on this data. For example, suppose that we want the wallet software to display a price chart of a given artwork over time; the underlying data required to construct such a chart might be scattered across hundreds of separate trade tickets in the blockchain in addition to the original artwork registration ticket. Or, we may want to see a single view of all of the artwork registered by a given artist, the data for which would also be spread across multiple tickets.

Thus, in our architecture, the blockchain tickets really serve as an entry-point to the system: a one-time ingestion process independently performed by each node, in a purely peer-to-peer manner, and without trusting or relying any other party or authority. When first syncing up with the network's blockchain, each node will scan every block in chronological order, inspecting at every distinct TXID included in each block. If a given TXID references a stored ticket file, then the corresponding file would be parsed, and each field in the

ticket would then be validated using a variety of validation rules, and each digital signature would be verified against its associated public key to ensure the validity of the ticket.

At this point, the node has "abstracted away" the underlying information stored in the ticket and is thus free to reorganize that data in its own private database. For this purpose, Pastel uses the highly regarded and ubiquitous *SQLite* package. This is a natural choice, as it is included in the Python Standard Library, and it is extremely fast, efficient, and reliable. A very simple table schema is used to store the various bits of information about each stored ticket. Thus, we end up with various tables, each of which contains different data; for example, we might have a table of users, with the primary key being the user's Pastel ID public key. Another table might contain the image fingerprint vector for every registered artwork on the system, while another table might contain all of the trade tickets, with the primary keys there being the combination of the underlying art asset's TXID and the corresponding trade confirmation ticket TXID.

In short, the nodes–and particularly the Masternodes–will have a sensibly laid out relational database that contains all the relevant information needed to run the network (information which, critically, the nodes can parse and verify themselves directly from the blockchain), so that both raw values and derived quantities can be quickly and efficiently retrieved. Consequently, the inefficient and slow "database" of the blockchain, which is slow and inefficient as a result of its decentralized, trustless architecture, can be leveraged for its security benefits without forcing us to also use this system for our actual computations, where it would present endless difficulties and bottlenecks.

12 Off-Chain Storage System

The old saying, "a picture is worth a thousand words," has particular significance for the Pastel project: the simple fact is that image files–particularly high resolution, lossless-encoded (e.g., as PNG files) image files, which can easily weigh in at over 10 megabytes, with 30 megabytes or higher being typical. Of course, it's possible to store high quality images using much more efficient lossy compression, such as JPEG files, but these sacrifice quality and accuracy and are generally frowned upon in "archival" settings such as fine-art and graphic design. Compare this to the size taken up by text files– the full plain-text of the English version of *War and Peace* from Project Gutenberg, which takes up 3.4 megabytes but compresses down to just 1.2 megabytes using simple ZIP compression.

12.1 The Need for an Off-Chain Solution

Clearly, the image files that Pastel will need to handle are much, much bigger than a text field in an efficient ticket file, and so our strategy for storing data in the blockchain itself will never work here– it couldn't even hope to scale to thousands of users, and the transaction costs to store the images would be enormous relative to the value of the coins. So we can't store the images in the blockchain. But then what do we do, since the images are obviously the most important part of a visual art registry; after all, art is intrinsically a visual phenomenon.

We address this challenge in our architecture in two basic ways. The first is that we embed a special representation of the image directly into the ticket file itself, which is then stored in the blockchain. The representation we use– the image fingerprint vector–is particularly *discriminative* and *robust*, despite taking up just 8,064 numbers (many of them zeros). Unlike a file hash, which changes completely when any bit of a file is altered, the fingerprint vector is relatively stable–even when the image undergoes a variety of common transformations, such as cropping, resizing, adjustment of color/contrast/brightness, common "filters" used in image editing software, etc. And even if the fingerprint does change, the fingerprints of two related images will have enough tell-tale statistical similarities that we can readily discern the relationship by careful (but automated) comparison of the fingerprints.

The second way we address the challenge is by storing the actual image files themselves in a novel decentralized file storage scheme, which we have been referring to so far as the *off-chain storage system*. Recall that in a Bitcoin-like blockchain, every full-node possesses a full copy of all the data in the blockchain; while this is obviously the "safest" way to implement a decentralized system, it's not scalable for large files. Thus, the essential sacrifice we must make in an off-chain storage system is that each node can only store a small part of the sum total of image file data stored in the network. This general idea is known as *sharding*, and there are many different ways to go about it, each with its own security, efficiency, and reliability trade-offs.

12.2 Limitations of Existing Approaches

Like any big and important problem, decentralized storage has attracted many smart people who are attempting to solve it in a variety of ways. These competing solutions can be roughly broken down into two types: storage-specific crypto-currency projects, such as *SiaCoin* and *Filecoin*, and more generic distributed storage projects, the most prominent of which are *IPFS*

and *Swarm*. While a detailed review of these projects is beyond the scope of this discussion, taking into account the specific requirements and priorities of the Pastel project, none of these solutions seemed at all attractive.

For one thing, Pastel is not about file storage–it's about rare digital art. File storage is just something that we need to have in order to achieve the main goals of the project. At the same time, it's absolutely critical that it work reliably and cost effectively. From our perspective, all of the mentioned projects are overly complicated in terms of technology. But even more importantly, they seemed to get the crypto-economics wrong, at least for our use case.

At this point, it's important to point out that the file storage requirements for our project are somewhat constrained in that:

- We never need to modify or delete existing files, since that is not allowed in the system.
- We don't need image retrieval to be particularly fast or high-bandwidth, since only the owners of the rare art will be permitted to access the files.
- Furthermore, when a user purchases the rare art, their client software will download and store the full-resolution originals on their local machine; thus, users will generally only need to re-download the files if they are installing the software on a new device.
- The thing that we care about the most is that the system should be extremely reliable in terms of data retention. The promise we make to artists who use the network is that, once they pay the one-time registration fee, they never have to worry about their art files being lost–even in a bad scenario where large parts of the network go down permanently.

Thus, we don't need a generic solution or system. We are willing to make all sorts of other trade-offs in order to ensure the highest possible amount of reliability and security while keeping the system simple and attractive for users of the network.

12.3 The Importance of Permanence

This emphasis on data retention, which is at its core an exercise in designing *fault-tolerant*, high-reliability systems, informs all the key design decisions used in the Pastel off-chain storage system. We are really trying to think of the network as being something sacred– like an *e-Louvre* in the way that Bitcoin is *e-Gold*; just as nothing is left to chance in the Louvre's humidity management

systems and other preservation protocols, we want to ensure that even in 100 years, the world does not lose access to a single one of the art images entrusted to the Pastel network. Yet anyone who has used computers over the last decades has a horror story about a dead hard drive that robbed them of their precious family photos or college term papers. Indeed, the only reason we seem to have progressed past this stage of digital impermanence is the rise of cloud storage services, which are of little use in a truly decentralized system.

That is no coincidence. The only way to create reliable systems out of unreliable components such as spinning hard disks and solid-state disks that inevitably wear down and lose their storage reliability is to use lots of them in different locations so that, as long as the components don't all fail at once (something so unlikely that we can essentially ignore it), we will always be able to reliably retrieve the files we store. This is precisely what Dropbox, Amazon, Apple, and Microsoft do with the data saved in their cloud services, and it obviously works well. But this sort of redundancy, when done correctly, requires huge financial and technical resources.

12.4 Reliability in Spite of Decentralization

Now consider how much harder everything becomes when you make it decentralized. Dropbox has the huge benefit of controlling its own infrastructure. If one of their data centers needs to go down temporarily for some reason, their engineers can plan for this situation and thus avoid downtime. But what happens when you don't control the servers because no one is in control? This is of course the essential point of a decentralized system. To see how different this is from the centralized scenario, imagine that a very enthusiastic Dropbox user liked the service so much that he asked the company if he could run the Dropbox server software on his own personal machine. It sounds absurd, right? Of course Dropbox wouldn't allow this– all the servers need to be in their direct corporate control for a host of reasons. Not the least of which is, what happens if this enthusiastic user needs to use the computer for something else? Or what if he stops paying his power bill and the computer turns off?

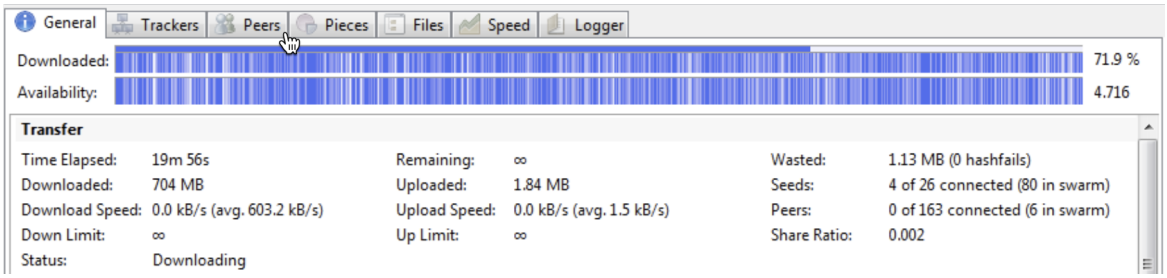
Yet, this is the exact situation that a decentralized MN crypto-currency network finds itself in. Just as a user does not require permission to set up a Masternode, the user doesn't need permission to disconnect it. Masternode operators are under no obligation to maintain the machines, and one has to assume that any given MN machine could instantly shut down– with zero notice to the rest of the users of the network, so that preparations can be made, such as copying the data to another machine. It seems almost a hopeless task to marshal a motley crew of computers run by random people around the

Internet into a data storage system that has the kind of permanence and reliability of a world-class art museum like the Louvre. Nevertheless, this is the problem we are faced with if we want to make a project like Pastel a reality.

12.5 BitTorrent and the Rare Chunk Problem

Most robust file storage/transfer system begins by splitting up the file into pieces. In the case of *BitTorrent*, the best known system of this kind, files are broken up into disjoint (i.e., non-overlapping) chunks. While BitTorrent generally works quite well for popular files, any frequent user of BitTorrent can tell you that torrents can get "stuck" at 95%+ completion because some of the chunks are unavailable on the network. Unless a user joins the torrent swarm with the complete file and "seeds" it, downloaders may never be able to get the entire file– a frustrating experience, especially for users on slow connections.

Figure 8: An Example of a BitTorrent Client Showing Disjoint File Chunks (Source: *µTorrent*)



The problem with BitTorrent is that you need to download every chunk to get the full file, and since these chunks are all disjoint, there will always be a "rarest" chunk that everyone needs but only a few users have. At the same time, other chunks will be plentiful, so that all the users can download them quickly. This is obviously a sub-optimal outcome. So what can we do to address this problem? One approach is to use what are called erasure-codes.

In an *erasure-code* scheme, there is some redundancy built into the chunks, which allows you to reconstruct some missing file chunks using the chunks that you do have. If this idea seems familiar, it might be because it's an old idea from the days of binary *Usenet* groups. Back then, users would post a large file broken into chunks along with several "PAR" (short for parity) files, which would allow for the reconstruction of missing chunks using special software. This is a good idea, and erasure-codes are a primary feature of such file sharing

crypto projects as SiaCoin. But given the specific design requirements for Pastel, we believe there is a better way, using what are generally known as *fountain codes*.

12.6 LT Codes Solve the Rare Chunk Problem

The first practical implementation of fountain codes was pioneered by *Michael Luby*, who ended up selling his software company to Qualcomm, the telecommunications IP giant. His ideas have since made it into pretty much every smart-phone since 3G came out. His implementation, known as *LT-codes* (short for *Luby Transform*), works by breaking up the data that you want to transfer–which could be a stream, such as a cell phone call on a digital cell network, or a generic data transfer in which the order of the data doesn't matter–into a series of chunks. The difference from BitTorrent is that the chunks in an LT-coding scheme are *fungible*. That means that any chunk is as good as any other chunk, as long is it is different from the chunks you already have. There is no such thing as a "rare" chunk because any new chunk you download will be useful in reconstructing the file.

This property is in fact what gives fountain codes their name: the analogy is that of a water fountain, with one or more "users" attempting to fill their water bottles at the same time. The users don't care which particular droplets of water they get from the fountain; they just place their bottles in the stream and hold it there until it fills up. Furthermore, the users don't need to synchronize their actions – one person could start filling a bottle, and then a few seconds later another person could come along and start filling their bottle – they don't need to wait for the next time the fountain starts up to get the stream "from the beginning". The way this works at a very basic level is that every chunk in an LT coding scheme contains certain random fragments of the combined file. It's almost as if someone took the entire file, put it in a bag, and then smacked it with a hammer, turning it into a bunch of oddly shaped fragments. Some of these fragments would be relatively large compared to the full size of the file, and others would be tiny bits of the file. Continuing this analogy, we could then reach into the bag without looking and grab a fistful of these fragments, which would constitute a chunk in the LT coding scheme.

Two things about this process are perhaps surprising. The first is that we can always figure out how to piece these fragments together. But perhaps this isn't so hard to believe, since if a porcelain vase fell and broke into a bunch of pieces, it might might take a while, but as long as you managed to not lose any of the fragments, these would only fit together properly in one way – the right way. We would start with the biggest fragments first, and then find the next

biggest fragments and try gluing them into place, and eventually piece together the vase. At this points it's important to realize the limitations of our analogy. In a real LT coding scheme, the bag of fragments would be bottomless, so that we could always reach in and grab a new fistful of fragments, each different from the previous ones, and each containing new pieces of the vase that would be useful in reconstructing it. At this point, the reader might be thinking that this process sounds pretty wasteful, since it would end up getting lots of duplicate fragments. Indeed, it's hard to see how this wouldn't lead to a lot of overhead in the protocol.

Obviously there needs to be some degree of redundancy/overhead in any robust system like this, but we should distinguish between redundant data and wasteful overhead. By wasteful overhead, we mean that some of the bits we are sending are not the actual data itself, but rather the protocol that describes the fragments and how to assemble them. While such information is needed to make the system work, we want to keep it to an absolute minimum. After all, if we had to download 30 megabytes worth of 3 megabyte chunks in order to reliably assemble a 5 megabyte file, that would be a pretty bad outcome, since the system would only be $5/30 = 16.6\%$ efficient. This leads to the second surprising property of LT codes: in practice, they can achieve efficiency rates of upwards of 85%, as shown in the following figure, which shows that in order to download 2,048 "symbols" worth of data with a high probability of success, we generally will only need to download 2,300 symbols worth of LT chunks. As we continue to download more chunks, the probability of successfully decoding the entire file approaches 100%.

This means that the vast majority of our bandwidth is being put towards useful work: transferring the bits of data we want, not useless overhead that uses up our bandwidth without getting us closer to what we want– reconstructing the file. And yet, we still get these almost miraculous benefits: we can get file chunks from multiple sources, in any order, and every chunk is just as good/useful as any other chunk. No more stuck downloads, no more "rare" file chunks that we hope someone will make available– just data droplets, like water from a fountain, that we can slurp up from anywhere until our glass is full and we have the file! But how will we know that we have the right file? That part is easy: we store the SHA3-256 hash of the original image file in the art registration ticket on the blockchain, and we keep getting new LT chunks until the reconstructed file hashes to this exact same value.

LT Codes were originally designed for a broadcaster-subscriber model, in which the broadcaster (for example, the NFL streaming a live game, or Steam serving a video game demo to users) has a full copy of the file/stream, and generates new LT chunks on the fly in an "endless stream" of chunks, and

Figure 9: Chart Showing the Efficiency of LT Encoding (Source: *Robust Scale-free Luby Transform Code and Its Performance*)

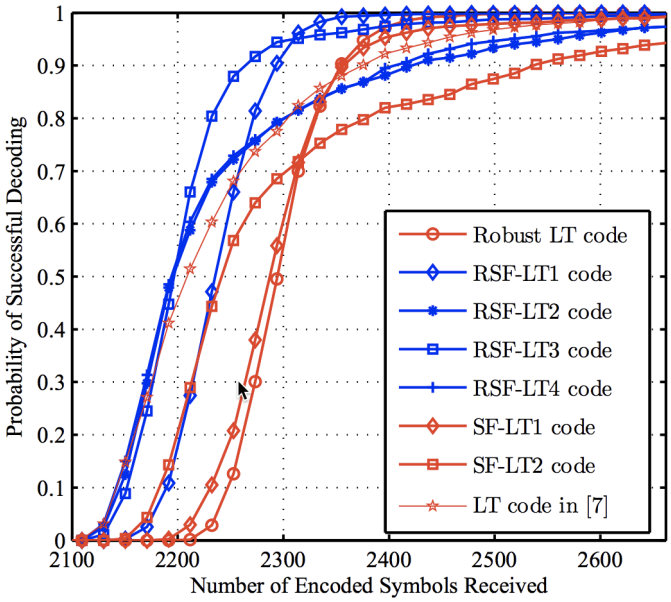


Figure 4: Probability of successful decoding versus the number of encoded symbols received. $K = 2048$ and $P_{era} = 0.1$.

the subscribers can tune in to the stream or begin downloading the file at any time. If you think about it, this would be impossible using a system like BitTorrent, since each chunk only contains a specific piece of the file; thus, all the subscribers would need to begin downloading at the same time in order for everyone to get the full file. One workaround might be for the broadcaster to have several streams going at once at different starting times, so that a subscriber could just wait for the next stream to start their download. But this obviously uses a lot more resources and is much less flexible and convenient for the subscriber. In any case, it is fairly straightforward to adapt this model to a more generic file transfer/retrieval context, in which several users on the network have varying numbers of chunks downloaded, and a few users have the entire file so that they can generate fresh chunks. Remember, our goal is to transform the art image file into a collection of LT chunks, and then to demonstrate that, even if we *randomly* delete or corrupt the majority of these LT chunk files, we can nevertheless reliably reconstruct the original file.

We must specify 2 parameters that control how the data is encoded into LT chunks; the first of these is the size of each chunk file. For this, we want to strike the right balance between the inconvenience of dealing with lots of small files, and the desire to be efficient in terms of not wasting bandwidth. That is, if the chunks are too large relative to the size of the data we are encoding, there could be wasted space from partially used chunks. Since most digital image files that would be registered on the Pastel network are probably on the order of 10 to 30 megabytes, we use a 2 megabyte *chunk size*. The other key parameter is the desired *redundancy factor*. What this controls is the total size of all the chunk files that we initially create. So if the original data in this case is 20 megabytes, and we use a desired redundancy factor of 12, we will create 20 x 12 = 240 megabytes worth of chunks; at a 2 megabyte chunk size, this means that we will start out with around 120 chunk files.

12.7 Chunk Replication and Self-Healing

The fact that we have many more LT chunks than would be required to reconstruct the image file is the first layer of protection for data stored in the Pastel network– it is what allows us to lose a large portion of the chunks while still being able to recover the files. The second layer of protection, which is equally important, is that *these chunk files themselves should be replicated*. That is, each distinct chunk file should be stored by several randomly selected Masternodes. This two-tier data protection scheme would give the network much more robustness and protect the art files from "Black Swan" type events, where huge portions of the infrastructure could vanish without warning. Such a risk is not just theoretical: for example, suppose that at some point in the future, the market price of Pastel coins were to drop significantly, leading Pastel Masternodes owners to liquidate their holdings and shut down their MN servers *en masse*. Thus, such scenarios are something we need to be prepared for if we want the system to be maximally reliable.

The final step to truly protect the art files over the long term is to make the system dynamic, with the capacity for self-healing. Even if a newly registered artwork begins with the desired redundancy factor (say, 12x), and each chunk is replicated across several independently controlled Masternodes, this redundancy could be eroded away over time as MN machines shut down and chunk files are deleted or corrupted. In order to combat this, MNs can be randomly selected by the network to maintain the redundancy level of each registered artwork; if an artwork is found to be below required redundancy, a randomly selected Masternode could reconstruct the original file and then proceed to generate replacement LT chunks, advertising the chunks to the other

Masternodes so they would be automatically mirrored by the network. In this way, the storage network is dynamic and *self-healing*: it can take a beating without dying, and will gradually recover to its initial strength, at which point it can handle another beating without ever losing the precious art image data.

In the next sections, we describe in detail how we implement both of these features in the Pastel system.

12.8 Distributing and Transferring Chunks

So far, we have introduced the LT coding scheme, and shown how it can be used to break up an image file into a set of fungible chunks. But that is only a starting point– we need some way to distribute all of these chunks across different Masternodes so that the data is distributed across the network. Therefore, we require a way for a Masternode to issue a request to the network for chunks related to a specific artwork (as identified, say, by that artwork's registration ticket hash, or by the TXID of the ticket), and we also need a way for the other MNs to transfer the relevant chunks to the requesting MN.

Clearly, there are many ways to go about doing this. The most obvious, naive approach would be to have each MN advertise a list of LT chunks, which could be identified by a pair of hashes: one hash would be the underlying artwork registration ticket hash, and the other would be the file hash of that particular LT chunk file. Then, other MNs could poll each other MN periodically to look for newly available chunks, so that each MN could keep track of which node has which chunks. The problem with this approach is that it generates a huge amount of network overhead: MNs would be constantly wasting bandwidth by advertising the chunks they have, and if you wanted to find a given chunk in the network with a high probability of success, you might have to connect to and check the chunk contents of lots of machines, making the system slow and inefficient.

Furthermore, the naive approach would still lead to the problem of determining which node *should* store each LT chunk– something that, for security purposes, we would prefer that MNs have no control over, since a malicious MN could use that to target all the chunks of a given file in a denial of service attack. And what would happen as MNs enter or leave the system? How would the "missing" files be replaced as old nodes shut down, which could happen at any time? We need some way to handle this "re-allocation" process for chunks as the underlying network size flexes up and down. These considerations lead us to the use of so-called *DHTs*.

12.9 Kademlia and Decentralized Hash Tables (DHTs)

It turns out that there is a brilliant invention that completely solves the distribution problem, and it was published in 2002 by David Mazières and Petar Maymounkov in their ground-breaking paper, *Kademlia: A Peer-to-Peer Information System Based on the XOR Metric*. With over 3,500 citations, this paper inspired whole new avenues of research in computer science and distributed systems. The original ideas have been expanded on, with a recent example being the 2014 dissertation *TrustedKad – Application of Trust Mechanisms to a Kademlia-Based Peer-to-Peer Network* by Michael Kohnen, which combines elements of reputation tracking with the core concepts of Kademlia.

So how does Kademlia solve all our distribution challenges? Notice the words "XOR Metric" in the title of the original *Kademlia paper*– this should seem familiar, since we use the same idea in the MN rank ordering method. It turns out that this simple idea, which provides a consistent "distance metric" that can be computed for any binary string of data (which ultimately can be used to encode anything, such as an alpha-numeric blockchain address or file hash), automatically eliminates the need for any of the architecture discussed above in the naive approach. That is to say, no wasteful communications about which node is storing which chunks, no complex or centralized system for deciding which node is responsible for which chunk, no checking through lots of MNs to find one with the relevant chunk, and no complicated logic for handling chunk re-allocation in the case of MNs entering and leaving the network.

So how do we get these miraculous properties? In short, here is how it works in the context of the Pastel network:

1. First, we observe that every LT chunk can be uniquely identified by its SHA3-256 hash, and similarly, every Masternode can be uniquely identified by a SHA3-256 hash– for example, we could create a *node identifier* by concatenating a MN's Pastel ID public key together with the corresponding MN collateral address on the blockchain, and then take the hash of the resulting string to serve as the identifier. **Now we have a unique hash value for each LT chunk and each MN**, both of which are simply hexadecimal strings of the same length.

2. If we take the binary representation of these strings– that is, the precise sequence of zeros and ones that the computer uses to store this data in memory– then we can easily **compute the XOR distance metric between them**. To do this, we step through each entry in the two binary strings, taking the next character of each and comparing them:

if they are both ones, then we write down a **1**, otherwise we write down a **0**. When we are finished stepping through the two binary strings, we take the resulting sequence of 0's and 1's that we just wrote down and interpret this as if it were the binary representation of an integer, which produces a simple whole number– the XOR "distance" between the strings.

3. **The larger this whole number distance is, the "further apart" the chunk is from the Masternode in the network**. At least, that's how we are defining it. Although this notion of distance is unfamiliar and strange, with no good analogy in the physical world (being so different from our normal conception of distance in the spatial sense), we have already seen in a previous section how the XOR metric satisfies the 3 requirements for a distance metric to be considered mathematically well-defined, so there are no real technical objections to using it in this manner.

4. **Finally, Masternodes are responsible for storing precisely those LT chunks that are "closest" to them in the network using this notion of distance**. As new chunks are added to the network, and as MNs enter and leave the network, this set of "closest" LT chunks to a given Masternode will change– perhaps dramatically so. For example, if a new MN is registered on the network, and the hash of the new MN's identifier is the "new closest" hash for certain LT chunk hashes, tyhen the new MN is deemed by the network to be responsible for storing the corresponding LT chunk file.

5. **Observe that we can also use this same method to compute a distance between MNs**, so that we can also sensibly discuss the set of "closest" MNs to a given MN in the context of the off-chain storage system.

And that's basically it– this little trick allows the network to, in a completely distributed, deterministic way, self-organize into a particular network topology. This is analogous to asking a large group of people to "hold hands with their neighbor" until they form a ring of people. Such a system is generally known as an *overlay network*, since it is "induced" on top of the underlying physical network that connects the nodes on the Internet. As a result, it makes no difference where a MN machine is physically hosted. For example, one MN could be in New York, and another two MNs in Tokyo, and the New York MN could still be "closer" to the first Tokyo MN than the second Tokyo MN.

Because hash functions generate what are essentially random outputs (i.e., each entry in a hash string is equally likely to be any of the permissible characters), the distances between LT chunks and MNs must also be randomly distributed across the set of MNs, with no structure or discernible pattern in terms of which MN is assigned to store which LT chunk. This is highly advantageous for the security of the network, since it makes it extremely difficult (if not impossible) for a malicious MN owner to impact or control which chunks are assigned to it– at least without the attacker controlling a large percentage of all MNs– since these are the result of a series of random accidents among strangers and their long-term identities on the network, not to mention the contents of random fragments of compressed data.

But by far the biggest gain is this: rather than needing to ask other network participants about where LT chunks are located on the network, we can now *compute* this information ourselves. Consider all the protocol overhead savings this implies: it gives us a way for all nodes to follow the same game plan without actively coordinating this plan, and also without any of them being in charge. So how do we go about computing this information? Well, we know the hash value for every valid Masternode (this list of current valid MNs is always available to us via the integrated RPC service in the underlying C++ blockchain code), and we know the hashes of the LT blocks that were used to store the image file corresponding to a given artwork, since this list of hashes is included in an artwork's registration ticket. Of course, though this ticket is stored in the blockchain, at this point the node would have parsed and ingested the ticket into its local database representation, making it a quick and easy database lookup to retrieve the list and use it for computations.

At this point, we simply need to perform a series of simple XOR distance calculations, which are trivial, and then the node can compile an internal data structure that allows it to keep track of all the relevant information. Thus if a Masternode wants to find out which MNs to ask for the LT chunks for a particular artwork, it would first get the list of required LT chunk hashes from the blockchain ticket for that artwork, and then, for each of the hashes, it would look up the XOR distance between that hash and each of the currently valid MN identifier hashes. Finally, it would then sort all the MNs by this distance number, and the MN with the smallest number would be the one responsible for storing that particular LT chunk (i.e., the chunk with that particular file hash).

If a new MN were to join the network, and coincidentally, happened to have a shorter XOR distance to a particular LT chunk than every other MN (something that would be fairly unlikely if there were hundreds of MNs), what would happen? The way the off-chain storage system is designed, this new

MN would automatically determine that it is the new closest MN for that LT chunk, at which point it would retrieve and store the chunk and begin serving it to the network– that is, sending it in response to valid requests from other MNs. Meanwhile, the MN that had previously been the "closest" MN to the chunk (i.e., before the arrival of the new MN) would also automatically determine that it is no longer the closest and realize that it is no longer required or expected to store that particular chunk anymore– thus it would soon delete the chunk to free up space for storing new chunks as they are created on the network, a process that, again, happens automatically.

12.10 Achieving Chunk-Level Redundancy in a DHT

We now have a way for assigning an MN to each LT chunk, which gives us a nice way to store and distribute chunks in a decentralized way. But we also want to have network redundancy in the form of storing the same LT chunk file multiple times on the network (so that it is available from more than one MN), rather than just relying on having many more LT chunks than would be required to reconstruct the original image file. How can we accomplish this if each LT chunk is mapped to precisely one MN at a time? The approach that we take is to compute *successive hashes* of the original LT chunk file hash. That is, we take an LT chunk hash– say, one that we retrieved from an art registration ticket– and compute the hash of this hash. This second hash, an example of what we call an *alias*, is also a secure, unambiguous representation for the LT chunk– indeed, it is just as good as the original hash. Thus we can sensibly use this alias to link a second MN to the LT chunk: the MN closest to the alias hash.

Taking this idea further, we can continue this process as many times as we like, creating a sequence of hashes that would all be linked to a particular LT chunk file, and which would each be linked to a different MN through the XOR distance metric ranking. In this system, the number of required aliases would be a network-level parameter, such as 10; this would imply that every individual LT chunk is stored by 10 different MNs, with the particular group changing over time as new MNs enter and old MNs leave the network. Importantly, this method preserves the compute/lookup property of Kademlia: we can still figure out–and agree upon–which MN to ask for a given LT chunk, without actively asking any other MN for this information or relying on any other MN to communicate this information to us.

12.11 Self-Healing in the Context of the DHT

This leaves the problem of the self-healing. Despite the redundancy introduced by the alias system, which is itself "self-healing" or self-balancing in that a replacement MN is automatically found when an old MN leaves the network. Yet it is still conceivable that a particular LT chunk could be lost forever: a very large and sudden drop in the number of available MNs– either as a result of market forces, or because of an attack on the network, or both– could wipe out all the MNs hosting that chunk before new MNs could rush in to take over from them. In this case, what could we do?

It turns out that each LT chunk is uniquely determined by two things: the first is the original data that we are turning into LT chunks, and the second is what is known as the *random seed* for a given LT chunk. The random seed is basically just a long, random integer that is generated when the LT chunks are first created. The set of these random seeds, together with the file hashes for each LT chunks, is also contained in the artwork registration ticket. This therefore gives us everything we need to implement self-healing: if MNs on the network determine that a given LT chunk is no longer available (i.e., it cannot be retrieved from the network using the file hash of the LT chunk or any of the aliases for that chunk), then the highest-ranked MN could first retrieve enough LT chunks so that it can reconstruct the original file, and then use the random seed corresponding to the missing LT chunk to generate from scratch the identical LT chunk. This process can be verified easily by computing the file hash of the "new" LT chunk and checking that it matches the file hash listed for that chunk in the original artwork registration ticket on the blockchain.

12.12 Implementation of the System

We employ a novel Python implementation for the off-chain storage system. For the LT chunk generation and reconstruction, we make use of code derived from the open-source Python LT Code implementation by Anson Rosenthal. The DHT code is also inspired by the open-source Python implementation of Kademlia by Brian Muller. For managing the actual file transfers over the network, we use the ZeroMQ library.

13 Serving Metadata and Thumbnails to Users

Many things that are simple and easy in the context of a centralized web application become problematic in the context of a truly decentralized system.

One example of this is the serving of various kind of metadata to users, such as low-resolution thumbnail images and information regarding usage of the network (e.g., which artworks have been viewed the largest number of times, or which artworks have appreciated the most in price in the past month). Essentially, we need a way of handling this data that doesn't fall into the categories of blockchain data (i.e., tickets stored in the blockchain) and off-chain storage (i.e., the image files themselves, which require that a requesting user proves ownership over the file in question).

To address this problem, we basically just use the same system as we use for the off-chain storage system, but with the difference that we do not require "authentication." That is, a brand new user (say, an art collector) on the Pastel network should be able to browse through the available content on the network before purchasing anything. Since this sort of data is relatively small in size, it should not place an undue burden on the MNs to serve this data to users. Of course, this introduces the potential for denial-of-service attacks, in which malicious users request a huge number of thumbnails in order to exhaust the network's bandwidth and computational resources.

As a result, before a new user can request such metadata, they must first register their identity by means of an identity establishing ticket on the blockchain. Thereafter, any such requests for metadata (which are made on a peer-to-peer basis to any valid Masternode by means of the Pastel messaging system) are signed with the requesting user's Pastel ID private key. This allows for MNs to track requests by each Pastel ID public key, and any users found to be making an unreasonably high number of requests over a given time-frame can be temporarily banned or blocked at the IP level by the Masternodes. Since it costs a non-trivial amount of Pastel coins merely to register a new Pastel ID, this makes it much harder and more expensive for a malicious user to try to attack the network by creating a huge number of Pastel IDs and using them to request the maximum amount of metadata.

14 Inappropriate ("NSFW") Content Filtering

As mentioned in previous sections, preventing unacceptable or illegal images from being registered is critically important to any sort of decentralized file storage system. It is well known by now that most existing systems, notably *FreeNet*, are used extensively to trade child pornography and other odious content. Not only do we have a moral obligation to avoid such an outcome, but we also have a legal obligation to MN operators, since even the passive hosting

of such content opens up a legal "can of worms" that everyone would want to avoid. Many decentralized systems take the easy way out by encrypting the stored file chunks. The thinking here goes that, since at worst, an individual node in the network is storing just a small encrypted fragment of illegal content, the operator of the node would not be held responsible for serving such content. But in the Pastel system, images– just like messages and tickets stored in the blockchain– are stored in an unencrypted format, which is important for a variety of technical and security reasons.

But even if this encryption issue were not a constraint, we believe that the issue of unacceptable content should be addressed head-on, rather than sidestepped through technical or legal loopholes. Thus we seek an *active* system for preventing unacceptable content from being uploaded to the network: that is, that we are better off "nipping in the bud" such content, rather than permitting any content at all to be registered and then relying on moderation or filtering for suppressing it. Fortunately, this same problem is faced by many other organizations operating on the Internet. For example, any website that allows users to upload a custom profile image that is publicly visible must ensure that users do not select an unacceptable image. Thus, open-source, robust systems are available for this purpose, including the one used by Pastel, which is called *OpenNSFW* (as in the phrase "not suitable for work") and was developed by the well-known Internet company, *Yahoo*.

These filtering systems tend to use machine learning in order to determine if image content is unacceptable. In order to develop such systems, vast collections of "inappropriate" images (as well as "appropriate" images) are collected and labeled with their relevant categories, and then *deep learning* models are trained using many GPUs over a period of many days to train the models, so that the models can "learn" to distinguish between the two categories. Other labeled images, which were not included in the initial training data, are then tested with the resulting model in order to determine how well the models generalized to new data that the models had never "seen" before. Essentially, new candidate images can then be tested on the trained model simply by supplying the candidate images as inputs to the model. This results in a single number between 0.0 and 1.0 being produced as output by the model, and this output can be thought of as the probability that the candidate image is NSFW: a number that is too high– say, like 0.98– is highly likely to contain inappropriate content.

So what do we mean by inappropriate anyway? The problem of rigorously defining what constitutes obscene material such as pornography is notoriously difficult; indeed, there is a very well-known United States Supreme Court case in which a Justice said that, while he couldn't clearly define such material in

words, "I know it when I see it." Of course, a case-by-case, manual test of the kind implied by his comment is not useful in constructing an automated, decentralized system , which is what we are trying to do in Pastel. Rather, we require a method that is deterministic and rigorous. Luckily, this has finally been made possible through the amazing advances in image analysis as a result of deep-learning methods. But deep-learning models are famously "black box" in nature; by this we mean that it is difficult to articulate in words what a particular model is doing on a particular input image. The intuition behind it, though, is fairly simple: the model learns to detect tell-tale markers of content such as nudity, for example, by looking for areas of the input image that resemble human genitalia.

The obvious objection at this point is that perfectly valid art can contain such content. For example, the renaissance masterwork, "The Birth of Venus" (shown in the following figure) contain body parts that might be detected as being "obscene" (e.g., female breasts/nipples) by an automated model; clearly, we would not want to forbid Botticelli from registering his painting on the Pastel network if he were still alive. Unfortunately, the current state of the art technology is unable to make such subtle determinations, and thus Botticelli's painting might be flagged as NSFW and prevented from being registered. While this is clearly sub-optimal, we believe that the downside from such "prude" over-sensitivity is a reasonable price to pay in the near-term in order to ensure that we prevent truly bad content (most notably, illegal child pornography) from being registered. The system Pastel uses for NSFW detection is modular, and the model can later be swapped out for a more powerful model in the future as new models are developed. Put differently, we elected to err on the side of caution for now, and, as we collect more data on how the Pastel system is being used in practice (for surely, creators of "artistically meritorious" content that is nonetheless deemed "pornographic" will make this known publicly), we can refine the thresholds used and re-evaluate our conservative posture.

What about unacceptable content that nevertheless manages to "get past" the automated detection system? For these, network participants will be able to make use of the built-in system for flagging unacceptable content to the network by means of an *Inappropriate Content Take-down Request* ticket. Such tickets, which can be validly generated by any Masternode, reference the specific image and include the reason why the image should be removed or suppressed by the network. Note that the underlying artwork registration tickets can never be removed, since they are written to the blockchain by means of coin transactions which are inherently immutable. Still, content can be effectively suppressed by the MNs collectively and *voluntarily* refusing to store

Figure 10: An excerpt from Sandro Botticelli's 'The Birth of Venus," painted in around 1484. (Source: *Google Art Project*)

the associated image data in the off-chain storage system, and similarly, the client software can suppress such artworks and their thumbnail pictures from being displayed to users in the wallet interface. In any case, once such a take-down request ticket is written to the blockchain, the MNs can vote by means of signed messages sent via the built-in messaging system as to whether they agree that reported artwork is in fact unacceptable. If a sufficient number of votes are gathered by independent MNs, the artwork in question can be suppressed in the manner described.

It's important that we be very careful with such "after the fact" regulation of the network content, since this is a huge potential security vector. For example, a malicious MN owner could report an enemy's valid artwork as being unacceptable purely out of spite. That's why it's critical for there to be a decentralized process in which the creator of the take-down request must, in effect, convince their fellow MNs operators of the correctness of the take-down request. MNs that are found to make spurious take-down requests– that is, if a large percentage of a MN's take-down requests fail to gather enough positive votes– could in the future be penalized by the network through the use of the reputation tracking system.

15 Near-Duplicate Image Detection

15.1 Introduction and Need for "Dupe Detection"

In the introductory chapter, we discussed how the fundamental design principle in the Pastel system is to build upon existing, proven technologies, and how the real contribution of the project is in its creative unification of various pieces of existing tech, which results in a product that is "greater than the sum of its parts." In the case of NSFW content detection, we were even able to leverage an entire existing system (*OpenNSFW*) and use it "off the shelf." Unfortunately, in the case of near-duplicate detection, this is not possible: there is simply no existing open-source solution for the accurate classification of candidate images that are, in some sense, "excessively similar" to an already-registered image. Therefore, we had to create our own; in many ways, the resulting "dupe detection" system is the single most innovative part of the Pastel software.

The need for a "dupe detector" is fairly obvious: the whole idea of a natively digital artwork being "rare" hinges on ensuring that this rareness continues; that is, the promise of rareness cannot be subverted– even by the original artist– without undermining the entire value proposition for collectors to engage in collecting rare digital artworks. Of course, one approach would be to say "Who cares if someone else who is not the artist registers a similar artwork? It's up to the collector to determine if a given artwork is genuine, and this is easy enough for them to do given the digital signature system used in the system." We believe that such a stance is counter-productive, since it deflects the problem onto users and exposes collectors to a variety of scams and other attacks designed by malicious users to trick them into purchasing a "rare" artwork that is not what they thought it was at the time of purchase.

It's easy enough to find examples of similar attacks in the wild in other contexts: for example, so-called "phishing" attacks often employ URLs that appear to be similar to a legitimate URL, but which in fact use similar-looking characters in the Unicode text encoding scheme to direct the victim to a fake website controlled by the attacker. Such an attack in the context of the Pastel network could be devastating to the public trust of the system: if an image that is very similar to a previously registered image (say, the identical image, but with the difference that the upper left pixel is made slightly brighter, so that file hash of the impostor image is not the same as the original image) were to be registered by an artist whose name also appeared superficially to be similar to the artist of the original image, collectors could easily be tricked and defrauded.

Again, the use of digital signatures in theory protects against this, since the cautious collector could verify the Pastel ID public key of the purported (say, by checking this on the artist's official website) and then verify that the signature of the purported artwork validates against this public key; so long as the original artist is able to keep their private key secure, no impostor artist would be able to forge or counterfeit this signature. But we believe that it's unreasonable to expect ordinary users, many of whom might not be technically sophisticated, to take such protective measures. Furthermore, this would still leave the problem of an unscrupulous artist, who, after creating a successful rare artwork with, say, 10 authorized copies, decides to register more copies of this artwork after the fact, perhaps with tiny superficial differences. Since this second version would, in effect, be the "same" artwork by the same artist, it would have the result of inflating the quantity of the artwork in circulation– something that is anathema to the goals of the Pastel project in that it undermines the very foundation of the value proposition: the reliable, continued rareness of a given artwork.

15.2 Approach and Technical Architecture

Now that we have established the need for a dupe detection system, and also explained that such a system was not available, at least in an existing open-source format, which is required for any decentralized system to be reliable, we will discuss how we went about solving this challenging problem. And it *is* very much a challenge, since there are an infinite number of ways in which an image could be modified in potentially subtle ways that nevertheless modify every single pixel value in the original image.

Often, when we approach questions of uniqueness in computer science, we can rely on file hashes. That is, if the hash of two candidate files is the same, then the files are probably identical (while it is theoretically possible to find two different files that hash to the same value– a so-called collision attack– this is impossible in practice when using modern hash algorithms such as *SHA3-512*, at least in a reasonable amount of time). But if we were to change a single pixel of the image– something that might not even be noticeable to a human observer looking at the image– the file hash would change completely. That's because file hashes are designed to be brittle in this way, a property known in the literature as the avalanche effect, so named because small difference accumulate just as a small amount of snow displacement on a mountain can result in the unleashing of a catastrophic avalanche.

What we need instead is a way to characterize an image (i.e., to take an image file as input and produce as output a shorter string or set of numbers)

that has the opposite property: that even as the image is modified in various ways, the characterization remains relatively consistent. That is, we want a *robust way to "summarize" the visual content of an image that is invariant under large classes of common transformations*. What do we mean by transformations? In short, this would be anything that a user could do to a given image (say, using software such as *Adobe Photoshop*) such that a typical human observer would deem the transformed image to be "essentially the same" as the original image– or at the very least, similar enough that it should be reasonably considered to be a "substantive duplicate" of the original image. The following list gives various examples of such transformations; the simplest and most straight-forward transformations are listed first, with the more sophisticated and complex transformations listed at the end:

- Modifying the image's brightness, contrast, saturation, blurriness, sharpness, or "curves."
- Cropping the image to show only a subset of the original image.
- Adding random "noise" pixels on top of the original image or using "dithering" to add a noise-like effect.
- Any kind of "linear transformation": that is, translation, rotation, reflection, linear scaling/stretching, perspective "skew" transforms, etc. of the original image.
- Various non-linear spatial transformations or "warpings" of the original image, such as non-uniform scaling, bending, twisting, bulging.
- Inverting or "solarizing" the original image.
- Adding a "lens flare" or similarly superficial addition to the original image.
- Chopping up the original image into disjoint pieces (e.g., the *Mosaic* filter in Photoshop).
- Edge-detection or "outline" filters that highlight sudden changes in neighboring pixel values in the original image.

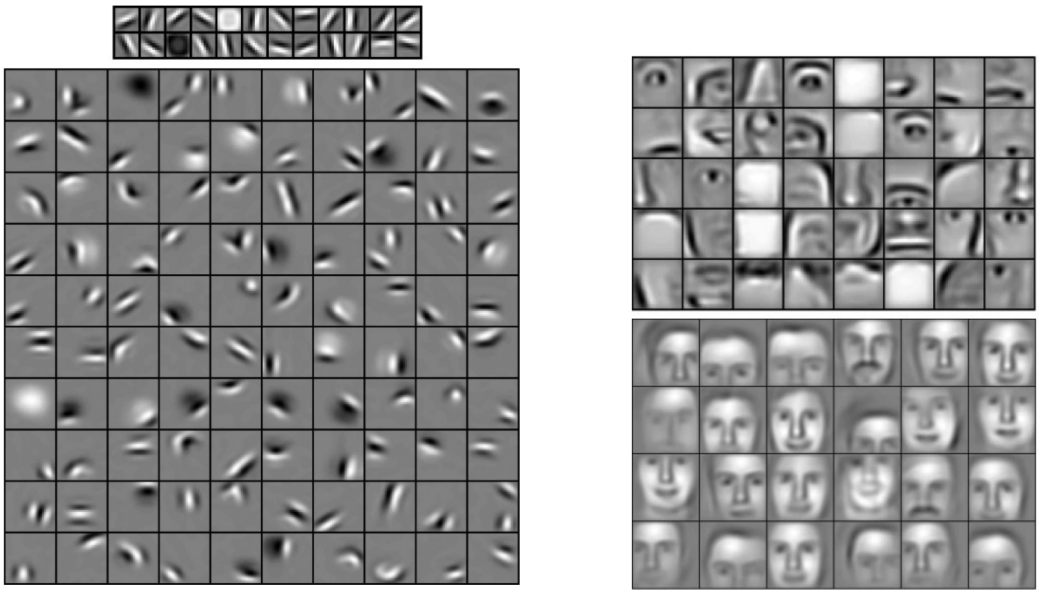
Any such list is necessarily incomplete, since the transformations can be combined in various combinations, and one could always come up with a slightly different way of transformation an image. Put differently, there is not a countable number of mappings or transformations from one image to

another, even if we were to limit the nature of the mappings that we want to consider. Thus, **any robust system for measuring the similarity between two images should make as few assumptions as possible about the nature of the transformations applied**. But since these transformations could result in every single pixel changing, it is hopeless to try to address this problem in the kind of simple-minded, systematic approach often favored in computer programming; for example, starting from the upper-left pixel of the first image and comparing to the upper-left pixel of the second image (and so on for the remaining pixels) is doomed to failure.

Luckily, we can again turn to the amazing power of modern deep-learning image models. Although a rigorous discussion of how deep-learning works is well beyond the scope of this document, the essence of how deep learning works on images is that it creates a hierarchical representation of the input image across various "layers," such that each successive layer is in some way "more abstract" than the previous layer. These layers are comprised of millions of "idealized neurons," which are simple computational constructs that take an input signal (which you can imagine as an electrical current of a certain strength), and if the signal exceeds a certain threshold, then the neuron "fires" by generating an output signal of a certain magnitude, which signal is then sent on to the set of other neurons that the given neuron is connected to. The computational implementation of these layers is simply a set of matrices describing *weights* (i.e., connection strength between neurons) and *biases* (i.e., how strong of an input signal is required to cause a neuron "activate" and generate an output signal).

For example, consider a face-detecting deep-learning model that is designed to output a "1" if an image contains a human face, and a "0" otherwise. The first layer of such a model might look at groups of pixels to see if they represent lines segments at various angles, as shown in the following figure. Then the next layer in the model might group these simple line segments together to form more complex curves, such as the "C-like" shape of a left nostril. The layer after that might combine these curves together to form something that looks like a nose or an eye. And then the last layer might combine these various parts– eyes, nose, lips, mouth, etc.– into a full face, as shown in the same figure. Once we have this more abstract representation, doing any sort of calculation, such as estimating if two photos of a face actually represent the same person, becomes much easier, since we are now able to reason at this higher and more appropriate level of abstraction: for example, we can now ask "are the eyes similar?" or "are the noses similar?" or "is the ratio of the distance from the eyes to the mouth versus the eyes to the nose similar?" and thereby estimate if the faces might belong to the same person. What is

Figure 11: An Illustration of the Increasing Levels of Abstraction Used in Deep Learning Models (Source: *Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks (DOI:10.1145/2001269.2001295)*)



particularly remarkable is that the human brain does much the same thing: for example, what neurologists call V1, the first layer of processing of the raw electrical signals generated by the human retina in response to a given stimulus, closely resembles the "line segment" layers found in deep-learning image models.

What makes deep-learning so powerful is that this abstract, hierarchical representation– which is so efficient and well-suited for various computational tasks– is *learned automatically from training data*. By this we mean that we don't have to go through millions of images and draw an outline around the nose, the eyes, the mouth, etc. We don't even need to articulate the existence of named features such as eyes or noses. Instead, we simply gather together a huge number of images of faces, which we might give a label of "1," and a huge number of images that do not include faces, which we could give a label of "0," and then run the model training algorithm. Training a model, despite seeming almost magical on the surface, is actually surprisingly simple: essentially, we

create the "skeleton" of the model by defining the number of layers, and how these layers link together using *activation functions*, and then initialize the millions of parameters of the model (i.e., the weights and biases in the neural network) with random numbers. At first, since this model is random, it will do a terrible job of classifying which images contain faces and which do not (since we have already labeled the images, we know what the right answers are and so can measure how good of a job the model does in predicting these labels)– that is, the *error* of the model when given certain inputs.

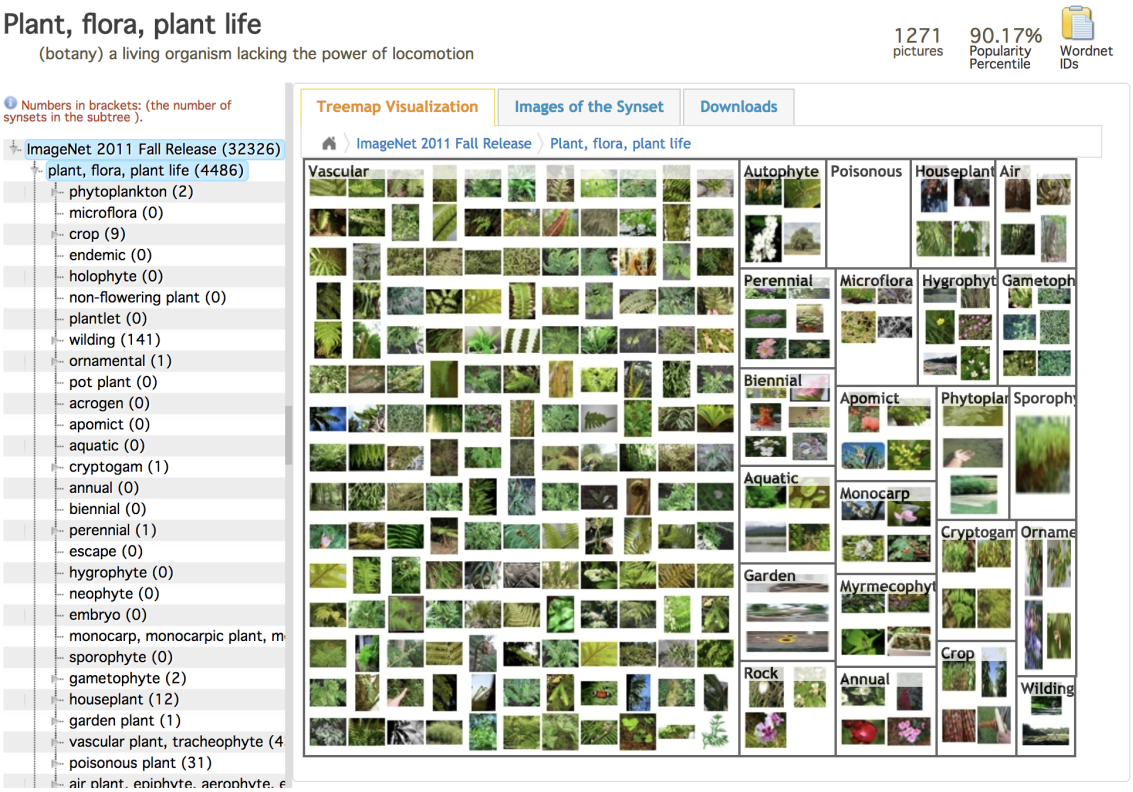
At this point, we just have an ordinary (if big and complex) mathematical optimization problem: how can we minimize the classification error of the model by changing the weights and biases. Such problems, though they may have historically been intractable due to limitations in computing power, are now very much solvable using a variety of techniques, the most common of which is known as *stochastic gradient descent* or SGD. The weights are all "nudged" in such a way that, after nudging them, the model now has a slightly lower error rate than before. By repeating this process millions of times, the model gradually "learns" an efficient representation of the underlying data which is well suited for computation. Thus we "automatically" end up with a compact, hierarchical model of the data that gets to the essence of whatever is contained in our training data– at least to the extent that such structure exists. That is, a model trained on random noise images won't learn anything useful, because by design there would be no structure to be learned.

Fortunately for us, many brilliant researchers have used these methods to train highly sophisticated deep-learning models that can classify and recognize the contents of real-world photographs. You can see these models at work in consumer products such as *Google Photos*, which is now able to automatically recognize people and pets in your photo collection. These researchers submit their models in rigorous, open competitions to compare the accuracy and reliability of different models that make use of various kinds of underlying architectures (i.e., number of layers; the "geometry" or relationship between the sizes of the various layers; the activation function used; the use of "pooling" or convolutional transformations between layers; the use of "dropout" to regularize the models to prevent "over-fitting," etc.)

Perhaps the most famous such competition known as ImageNet, in which competing research teams are given millions of labeled images as training data, and must train their models to predict the actual labels for a set of images which are withheld by the competition organizers until after all contest submissions have been received. The labels for the images span a very large number of categories, from animals to vehicles to foods. Furthermore, the categories can are highly nuanced; for example, there is a separate category for "Stingrays"

and "Manta Rays." The following figure shows an example of such training images and their associate sub-categories (the category illustrated in the figure is "plants, flora, plant life"):

Figure 12: Example Images From the ImageNet Competition (Source: *www.image-net.org*)



Since ImageNet debuted in 2009, there has been an enormous increase in the classification accuracy rate achieved by the winning models. Various teams, ranging from the University of Toronto and MIT, to private companies such as Google and Microsoft, have come up with different approaches. The great part about this is that these models, which are trained at great expense (the training process uses hundreds or even thousands of high-performance GPUs, which are run for days or even weeks at a time on millions of labeled training images), are then made publicly available so that anyone can use them. The most famous and popular of these models are made conveniently available in the popular Python package, *Keras*, which is essentially a "wrapper" built on top of Google's popular *TensorFlow* neural network architecture. The dupe detection system in Pastel makes heavy use of these models. In particular,

we use the following "named" models (the links for each model point to the academic paper in which the model was first introduced):

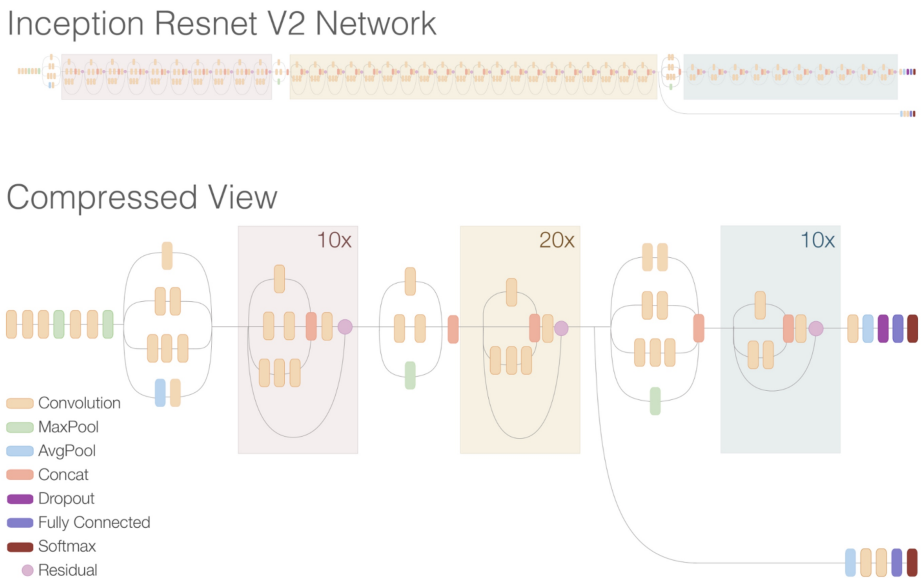
1. VGG19
2. Xception
3. InceptionResNetV2
4. DenseNet201
5. InceptionV3

While all of these models could be termed "deep learning models," they each vary in the particulars– in some cases dramatically so. To give a sense for what those particulars look like, the following figure shows a graphical depiction of the architecture used in the *InceptionResnetV2* model. Even still, they each have proven their merit through real-world performance in a challenging and objective contest, and thus are all worthy of consideration. Indeed, the fact that each model works in different ways in a very good thing, since a central tenet in machine learning and data science in general is that an *ensemble* of models– so long as each model works well on its own and the models are sufficiently different from each other in design and operation– outperforms any single model individually. This can be thought of as the "robotic analog" to the concept of "the wisdom of crowds."

So how is any of this relevant to near-duplicate image detection? Consider that each of the models above takes a color image file as input and then generates a vector of numbers as output. For instance, the first model above, *VGG19*, generates a vector that contains 512 numerical entries; another model generates a much longer vector containing nearly almost 2,000 numerical entries. These vectors, which we refer to as *image fingerprint vectors*, are how the models relate the input image to the set of potential label categories. Many of these entries will turn out to be zeros; the particular entries that are non-zero, and the magnitude of these in both a relative and absolute sense, are what "encodes" the outputs or "conclusions" of the model when given the image as input.

Although we are not trying to determine whether an art image file depicts a horse or an airplane or plant, it turns out that the generated image fingerprint vectors provide us with an ideal way of robustly characterizing an image so that it is relatively invariant under common transformations. By this we mean that, if we took a photo of a dog and generated the image fingerprint vectors

Figure 13: Graphical Depiction of the Network Architecture Used in the *InceptionResnetV2* Deep Learning Model) (Source: *Google AI Blog, 8/31/2016*)



from each of the models above for this image, then the resulting fingerprint vectors would be "similar" to the fingerprint vectors generated by taking that same image of a dog, but first modifying it– say, by stretching it, or adding random noise to the image. After all, whether or not a photo depicts a dog shouldn't depend on the exact placement of the dog in the photo, or the contrast/brightness of the photo, or even the lighting used when taking the photo: it should be robust to these superficial difference if it is to have any hope of generalizing well to new images not seen during training.

You can probably guess where this is going: for each image submitted for registration to the Pastel network, we first compute the fingerprint vectors for that image (after resizing the image to a resolution of 225 x 225) for each of the 5 models listed above. We concatenate all of these vectors into a single *combined image fingerprint vector* consisting of 8,064 numerical entries. Put differently, we end up with a list of 8,064 numbers which characterize the "opinion" or "takeaway" of all 5 deep-learning models when they are shown that image as input. Because each model is built using a different architecture that in turn learns different kinds of features, this characterization is more reliable than using any individual model in isolation. This is important, because there is a growing body of research literature detailing how individual deep-learning models can be vulnerable to being "tricked" given artfully constructed "fake"

inputs. For example, an image file that appears to a human to be random noise might be used to fool a single deep-learning model into thinking that the image shows a dog. But such "attacks" must be crafted with full knowledge of the architecture used in the model, so as to exploit the shortcomings of the architecture: by combining multiple different models, we make such trickery infinitely harder.

If the above explanation is still unclear, a simple (if macabre) analogy might be helpful: imagine that we have five adult humans that we are conducting an experiment on. Each of them is strapped in a chair and forced to look directly in front of them at a screen that can display an image to them that takes up their entire field of view, while a huge collection of electrodes is plugged into their brains that can pinpoint exactly which neurons in their brains are electrically activated. We then show all 5 people the same image for 10 seconds, and record the data generated by the electrodes. If there are N electrodes connected to each person's brain, then we could represent the state of their brains by a vector containing N entries. These vectors are analogous to the image fingerprint vectors described above, and each deep-learning model is analogous to the brain of one of the 5 human participants.

Essentially, we would be measuring how each person's brain processed, recognized, and categorized each of the images. By "throwing out" the raw image data and instead focusing only on the electrical measurements of brain activation, we are able to "piggy back" on all the neural processing power that the brain can marshal in evaluating visual stimuli. Generally, the human brain operates on the level of compressed representations that leverage the inherent hierarchical abstractions present in structured data such as natural images (i.e., as opposed to images of random noise such as a television displaying "static"). This compressed representation is inherently robust to superficial differences in things like lighting, orientation, scale, etc. That is to say, the measurements that are produced by the participants' brain activations (or at least their relative relationships/structure) will be relatively stable as the displayed image is transformed in various ways, such as by rotating the image slightly or changing the contrast or brightness of the image. Since deep-learning models like the ones we are using are based on similar principles to how human brains operate (at least for tasks such as visual perception; clearly, at present such models are incapable of general intelligence of the kind displayed by humans), this analogy is particularly apt.

15.3 Measuring the Similarity of Fingerprint Vectors

So now we have shown how we can transform any submitted image into an ordered list of 8,064 numbers, many of which are zeros. How can we now use these to spot near-duplicate images so that we can prevent them from being registered on the Pastel network? Just as we could never hope to compare the underlying images by comparing each respective pixel in the images, we can't take such a naive approach in comparing the fingerprint vectors. Although we said that the fingerprints are robust under transformations, this was really more on a relative basis when comparing them to a naive approaches such as a file hash. In reality, stretching or bending an image could result in every number in the fingerprint changing, with certain entries potentially changing in a dramatic fashion. Thus a simple comparison of the fingerprint vectors for two candidate images– let's call them **Image A** and **Image B**– is not going to work. That means that we can't, for example, compute the *Euclidean distance* (i.e., the ordinary mathematical notion of distance used in geometry for computing the distance between points in N-dimensional space, as in the the *Pythagorean Theorem* in the case of 2-dimensional plane geometry), and then say that "if this Euclidean distance is closer than a certain threshold, the images should be deemed to be duplicates of one another." That is, while we certainly could try doing it this way, it turns out that it doesn't work well in practice.

Thus we need to find a more clever way of testing two image fingerprints for similarity– one that is robust to large, non-linear changes in certain entries in the vectors. We really want a method that uses a statistical approach that deems two images to be similar if there are "too many coincidences" when comparing the size of the respective fingerprint vector entries than would be expected if the images were truly unrelated. That is, we need something that looks at the "shape" of the vectors– both the absolute size of the entries but also the relationship between these magnitudes across the various entries. We also want a method that can be computed "in a vacuum"; by this we mean that we want a method that can tell us the similarity between any two given images while only making reference to those two images.

This is important because we never want to have to do any computations on the entire set of registered image fingerprints at once, since this would lead to insurmountable scaling problems as the set of registered images grows. The ability to compute image similarity in a piecemeal, pair-wise fashion, makes the problem susceptible to a "divide and conquer" scaling approach so that, down the road, the dupe detection problem can be broken up into a large number of simple sub-problems, which can then be "farmed out" to various

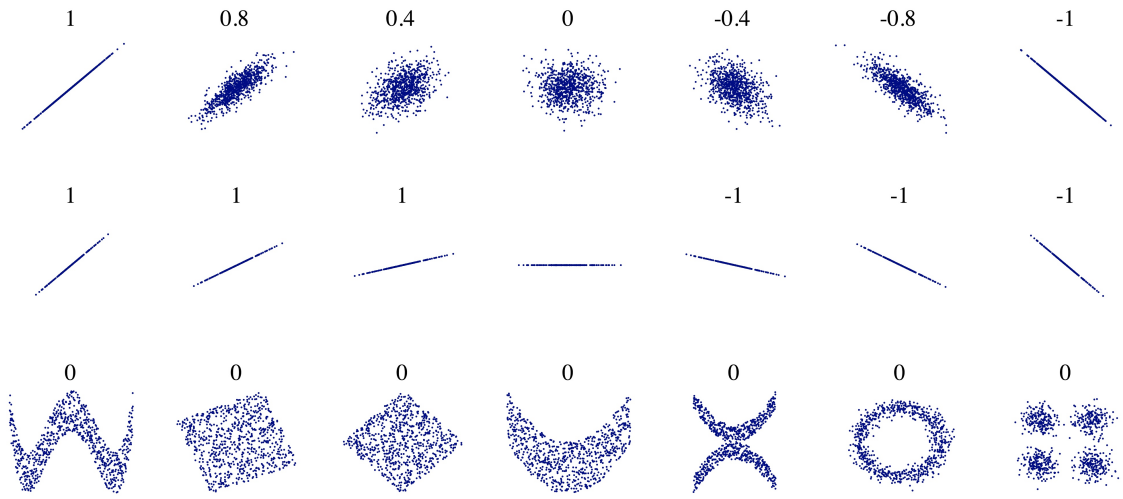
nodes in the network.

Clearly, measuring the similarity of two equal-sized vectors of numbers is not exactly a novel problem: any time you talk about two vectors or time-series being *correlated* (for example, the series of changes over time in the price of gold and silver, or a series of data points showing the elevation of a particular point in space along with the temperature at a given time), you are essentially asking the same kind of question. So what is correlation? According to *Wikipedia*, "In statistics, dependence or association is any statistical relationship, whether causal or not, between two random variables or bivariate data. In the broadest sense correlation is any statistical association, though in common usage it most often refers to how close two variables are to having a linear relationship with each other." When people usually talk about correlation, such as in the field of psychology (where a study might attempt to measure, for example, the correlation between anxiety and sleep deprivation) context, they are almost always referring to what is known as *Pearson's correlation coefficient*, also known as Pearson's r. But this is not the only way to define correlation.

At this point, rather than get bogged down in complex mathematical terminology and notation, we will take an intuitive approach in explaining how all this works. First, imagine a *scatter-plot* of two data series such as the ones depicted in the following figure. To make things concrete, suppose that the first series is the image fingerprint vector of Image A, and the second series is the image fingerprint vector of Image B. Then the scatter-plot would simply consist of a series of points in the plane; the first such point would have as its X-coordinate the first entry in the image fingerprint vector for Image A, and the Y-coordinate of that same point would be the first entry of the image fingerprint vector for Image B. We would then continue adding points to the coordinate plane, with one point added for each of the 8,064 numbers contained in each fingerprint vector. Consider the shape of the graph of the resulting "cloud" of points: if this graph appears to have no structure– that is, if it looks like random noise, similar to the circular "noise" pattern caused by firing a shotgun at a wall at close range– then the fingerprints can be said to have a low degree of correlation or dependency. An example of such a "zero correlation" scatter-plot is shown in the center of the figure in the top row.

Now consider the case where the scatter-plot shows a distinct pattern or structure, so that points are more concentrated in the upper-right and lower-left quadrants of the coordinate plane. An example of such a plot can be seen in the figure on the top row, second from the left. Such a pattern would be indicative of correlation between the fingerprints. Now imagine that we plot the best-fit line passing through the set of points in the plane– that is, the

Figure 14: Examples of Various Kinds of Scatter-Plots, Along With Their Pearson Correlations (Source: *Wikipedia*)



line that minimizes the "squared error" between the points and the line. The standard measure of correlation introduced above, Pearson's Rho, essentially tells us the slope of this best-fit line.

Intuitively, a positive slope implies that when a certain entry (say, the 6,024$^{\text{th}}$ entry) in the fingerprint vector for Image A is a large number, the 6,024$^{\text{th}}$ entry in the fingerprint vector for Image B will *also* tend to be a large number. Similarly, small entries in the fingerprint for Image A will generally correspond to small entries in the fingerprint for Image B. Notice that we said "tend" and "generally" in the previous sentence: this is because this isn't strictly true for each entry, otherwise the associated scatter-plot would look more like the upper-left plot in the figure– a straight line, rather than a "fuzzy" cloud of points showing the rough outline of a line at a certain angle.

The problem with Pearson's Rho, and the reason why it doesn't work well in our application of duplicate detection, is that it implicitly assumes that the relationship between the two series can be effectively modeled as a linear function, which is a particularly restrictive condition. In the real world, with all its complexity, very few things are truly linear in nature. Most natural phenomena, such as billowing smoke or the heart-rate of an animal, exhibit strong non-linear characteristics. This is certainly the case when it comes to the image fingerprints we are using, and particularly true when it comes to how these fingerprints change as the underlying image is modified in various ways.

This drawback is clearly visible in the bottom row of plots in the above

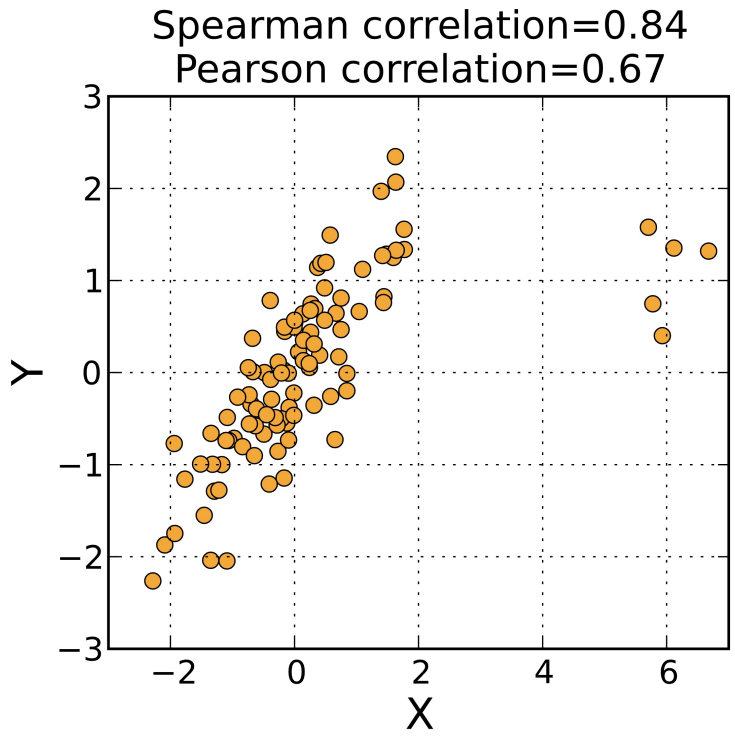
figure. Despite there being an obvious, structured relationship (at least, obvious from the standpoint of a human observer looking at the plot), all of these plots in the bottom row have a Pearson correlation of 0, implying that there is no correlation relationship between the series. Obviously, this isn't right. The problem is that we are using the wrong tool to measure correlation– one which makes assumptions about the nature of the relationship between the two series (i.e., that the relationship is essentially linear in nature). We need a measurement that will highlight when there is a any kind of "structure" in the scatter-plot. That is, we want a measure that will produce a number between -1.0 (completely negatively correlated) and 1.0 (completely positively correlated), and that will only lead to a number close to 0 if the scatter-plot looks truly random, like a shotgun firing pattern.

Basically, we want to relax our assumptions about the nature of the relationship between the two series– in our case, the two image fingerprints. The first measure that we will introduce for this purpose is known as *Spearman's Rho*. Unlike Pearson's correlation, Spearman's Rho does not assume that the relationship between the series is linear, but rather that it is *monotonic*, or "order preserving." For example, the well-known exponential curve, which looks somewhat like a hockey stick, is an example of a monotonic function that is not linear. Obviously, the exponential function is not a line, since it is curvy. Yet the number 2^3 (8) is larger than 2^2 (4), and this property extends to all numbers on the curve. This is a less restrictive condition; not only does it apply to all lines, but it also applies to all sorts of other curves. Sometimes, Spearman's Rho can end up being quite close to Pearson's correlation when computed on the same underlying data, but other times these measures can lead to very different results, as shown in the following figure.

The way Spearman's rho works is by first replacing the raw data points with their *rank orderings*. That is, if the largest entry in a given image fingerprint were, say, 7.124, we would instead replace that with 8,064, since that is the total number of entries. The median entry in the fingerprint would be replaced with 4,032 (half of 8,064), since this counts the number of entries that are less than the median. By throwing away the specific magnitudes and focusing exclusively on the rank order relationship between entries, Spearman's Rho is much more robust to outliers (i.e., very large entries in the fingerprint vector) which can distort and render useless a measure like Pearson's correlation, which uses the actual magnitudes of each entry in the vector. This explains why the plot shown in the above figure shows a much higher Spearman correlation (0.84) than the Pearson correlation.

The next measure of correlation that we will use is known as *Kendall's Tau*. Like Spearman's Rho, Kendall's Tau is also based on using the ranks

Figure 15: A Comparison of Pearson and Spearman Correlation (Source: *Wikipedia*)



of the data points rather than their magnitudes. However, it is computed in a different way that takes into account the number of "concordant" and "discordant" pairs. To understand what this means, suppose we select two random points in a given scatter-plot, the first point with coordinates (X_1, Y_1) and the second point with coordinates (X_2, Y_2). These two points are said to be concordant if the sign (i.e., positive, negative, or zero) of X_2 minus X_1 is the same as the sign of Y_2 minus Y_1. A discordant pair would be when the signs are different.

Essentially, Kendall's Tau is "proportional to the difference between the number of concordant pairs and the number of discordant pairs" (Source: Wikipedia), where these concordance calculations are performed on the rank orderings of the entries rather than their raw magnitudes. In practice, Kendall's Tau is fairly similar to Spearman's Rho, although it is considered to be more powerful and unbiased, but this comes at the cost of significantly greater computational intensity for Tau versus Rho. Using the "Big O notation" common in computer science, Kendall's Tau has complexity $\mathcal{O}(n^2)$, while Spearman's

Rho has complexity $\mathcal{O}(n \log n)$. Both Kendall's Tau and Spearman's Rho are famous and popular correlation measures, and efficient implementations of both are included in the Python packages *Numpy* and *Pandas*, which are used in the Pastel software.

Unfortunately, even these more powerful measures of correlation are limited: Tau and Rho still assume that the underlying relationship between the series is monotonic, which may or may not be true. What we would really like, in the final analysis, is more of a measure of *statistical dependency*– one that doesn't make any assumptions whatsoever about the nature of the relationship between the series– or even assumes that the relationship takes the form of a mathematical function (i.e., that there is a unique association between each input and each output, which precludes the situation where the same input could be mapped to two different outputs). It turns out that this problem was completely solved in a 1948 paper by *Wassily Hoeffding*, A Non-Parametric Test of Independence. Hoeffding called his measure of dependency introduce in this paper $\mathcal{D}$. The actual computation of Hoeffding's $\mathcal{D}$ is quite involved and complicated. Suffice it to say, like Rho and Tau, it takes two series as inputs, and then generates an output number between 0.0 (totally statistically independent) and 1.0 (totally statistically dependent).

Hoeffding's $\mathcal{D}$ is somewhat miraculous in that it makes absolutely no assumptions about the underlying relationship between the two series. There could be some hideously complex, highly non-linear mapping between entries of the first series and entries of the second series, but that doesn't concern us at all: any dependency between the series, no matter its structure, will be reliably detected by Hoeffding's $\mathcal{D}$ metric, almost as if we had "X-ray glasses". This makes it a perfect candidate for us in terms of measuring the similarity between image fingerprint vectors, since it allows us to approach the problem without any preconceptions concerning how the two images or their fingerprint vectors might be related.

So why not just used Hoeffding's $\mathcal{D}$ for everything and forget Spearman's Rho and Kendall's Tao? The short answer is that, if we could, we would; the problem is that Hoeffding's $\mathcal{D}$ is almost comically difficult to compute as the number of samples in the series (i.e., the number of entries in the fingerprint vectors) increases. Furthermore, unlike the much more popular and common Tao and Rho, there is no highly optimized, freely available implementation of Hoeffding's $\mathcal{D}$ that is exposed as a Python library. Indeed, the novel open-source implementation of Hoeffding's $\mathcal{D}$ is the first open-source, Python implementation of the algorithm in existence. But despite the use of high-performance *Numpy* matrix primitives in this implementation, computing $\mathcal{D}$ for two vectors (each consisting of 8,064 entries, as our image fingerprints

do) takes almost an entire minute on a 16-core, modern machine. And this is just for a single comparison– in practice, we would need to compare each candidate image fingerprint vector to thousands of previously registered image fingerprint vectors. Thus, we obviously can't use this approach since it would be infeasible. But rather than give up and make do with the simpler, easier-to-compute measures, we instead take a multi-prong approach to make the problem tractable:

1. First of all, we do not attempt to apply Hoeffding's $\mathcal{D}$ to every registered fingerprint vector. When two fingerprints are either very similar or completely dissimilar, both Spearman's Rho and Kendall's Tau do a very good job of accurately determining whether or not the second image is a near-dupe of the first. It is only when the two fingerprints are in a "gray area" in the middle that the superior power of Hoeffding's $\mathcal{D}$ is required. Thus, we first use Rho and Tau (which can be computed quickly and efficiently) to eliminate the vast majority of previously registered fingerprints. When the Rho or Tau is above a certain threshold (specifically, 0.80 for Kendall's Tau and 0.86 for Spearman's Rho– found empirically through trial-and-error), then there is no need to compute Hoeffding's $\mathcal{D}$, since the fingerprints are almost certain near-duplicates of each other. Indeed, we take this principle even further and only compute Kendall's Tau if Spearman's Rho is below 0.80. In practice, over 95% of the previously registered fingerprints can be eliminated in this way, so that we only need to bring out the Hoeffding "big guns" for a small fraction of registered images.

2. Second, we take advantage of parallel computation extensively, breaking up the computation across multiple CPU cores in order to fully exploit the parallelism available in modern CPU designs. This provides another large boost in speed in the cases where we do need to do the harder computations.

3. Notwithstanding the first two strategies, which dramatically help, we are still not even remotely in the realm of feasibility if we assume a large number of registered images. Thus we need to find another way to slash the computational burden so that the approach is practical. We do this by introducing a novel "*bootstrapped*" version of the Hoeffding's $\mathcal{D}$ algorithm. This technique, which is not found in the literature, takes advantage of a powerful idea in statistics, known as the Bootstrap. Essentially, instead of using all 8,064 entries in each image fingerprint vector, we randomly select a certain number of entries from the vectors

with replacement, so that it's possible for us to choose the same entry more than once.

Note that we are really selecting the *indices* of the vectors, and then extracting the same entries from the two vectors. For example, if we selected the index 4,055 as our first index out of, say, 100 selected indices, then we would look up the $4{,}055^{\text{th}}$ entry in the first fingerprint vector and write it down, and then look up the $4{,}055^{\text{th}}$ entry in the second fingerprint vector and write it down. We would then repeat this process another 99 times until we have extracted 100 entries from both vectors. Then, we simply run the Hoeffding computation on these much shorter vectors, which contain just 100 numbers rather than the full 8,064 numbers; unsurprisingly, these shorter vectors can be computed nearly instantly.

At this point you might be objecting that one could hardly expect these 100 entries to accurately characterize the full 8,064 entry vector, and this is of course true. Here is where the "bootstrap" idea comes into play: we repeat this process, say 50 times, each time getting a different set of 100 randomly selected indices, and compute Hoeffding's $\mathcal{D}$ for each of these 50 "boostrap" samples. Thus we end up with 50 different estimates of $\mathcal{D}$, and although each estimate is inaccurate to a certain degree, it turns out that if we take the average of all 50 of them, we end up with a remarkably good estimate for the actual $\mathcal{D}$ that would be computed using the full 8,064-entry vectors.

In practice, we try to be clever here as well, so that our average estimate is not unduly effected by an "outlier" estimate for $\mathcal{D}$ (i.e., if one of the 50 estimates we generated happened to be very different from the rest because of some freak coincidence, we don't want that to distort the overall average). Thus, we first order all of our estimates for $\mathcal{D}$ and then throw away any of these that are below the 25th percentile and above the 75th percentile. Finally, we take the simple arithmetic average of the remaining estimates for $\mathcal{D}$. This shrinks the required time to perform the Hoeffding's $\mathcal{D}$ computational sufficiently for it to be feasible even for a large number of registered fingerprints.

15.4 Putting It All Together

At this point, the problem of determining whether a newly submitted candidate image is a near-duplicate is straightforward: first, we compute the fingerprint vector for the candidate image. Then we go through the set of

fingerprint vectors for all previously registered artworks in the Pastel network, which are conveniently stored by each Masternode in a local *SQLite* database. While these same image fingerprints are also contained in the original artwork registration tickets stored in the blockchain, the database representation is much faster and easier to work with for the kinds of computations we are doing here, so we use the database instead of the raw blockchain data for these purposes.

Then, for each registered image fingerprint vector, we compute the Spearman's Rho between that vector and the fingerprint vector for the candidate image. If the resulting Rho is above our threshold of 0.86, then the candidate image is deemed to be a duplicate and the registration is not permitted to move forward. If the Spearman's Rho is below our threshold of 0.86, we then compute the Kendall's Tau between the fingerprints. If the resulting Kendall's Tau is above our threshold of 0.80, then we deem the image to be a duplicate and prevent it from being registered. Finally, if the Tau is below our threshold of 0.80, we compute the bootstrapped Hoeffding's $\mathcal{D}$ metric described above. If the resulting $\mathcal{D}$ metric is above our threshold of 0.48, then the image is considered to be a duplicate and cannot be registered. Otherwise, the image is deemed to be original and is permitted to proceed with the registration process. Of course, in order to finish the registration process successfully, the candidate artwork must also pass the NSFW detection test and all other required conditions.

So how well does this all work in practice? In our experiments using real-world images, in which we prepared intentionally derivative images that could be spotted as dupes by a human observer as well as genuinely "new" images that had never been registered, we found that the approach showed a greater than 90% accuracy rate in detecting both duplicate and non-duplicate images. So that readers can evaluate these claims for themselves, we have prepared a "self-contained demo" illustrating how this all works, together with links to download the test images we used. This demo is a single Python code file that can be run in relative isolation to the rest of the system. A more modular, readable version of this same code is used in the actual Pastel software to implement the dupe-detection functionality.

The end result is, in our view, by far the most powerful and discriminative near-duplicate image detections system available anywhere today, at least as an open-source project. And it's a good thing, too, since such functionality is absolutely critical if we want our system to be truly decentralized and automated– that is, not reliant on subjective human judgment and moderation. Duplicate detection– when done correctly, so that we do not end up with a large number of false positives (i.e., where we deem a candidate artwork to

be a duplicate when in fact a reasonable human observer would classify it as an original artwork) or false negatives (i.e., where we deem a candidate artwork to be original when in fact it is a duplicate or closely derived from an existing registered artwork)– is what enforces true rareness in the system. Not only is a specific artwork guaranteed to be rare (i.e., the artwork as referenced by the TXID of its associated registration ticket on the Pastel blockchain), but the specific underlying image itself (or rather a robust, flexible interpretation of what we really mean by a "specific image" that includes any image obviously derived from the original image) can only exist a single time in the system. Not only does this protect artists against other users ripping them off with obvious "clones" of their registered artworks, it also protects collectors of artwork, since it gives real "teeth" to the notion that a given image, once registered, cannot be registered again by anyone else– even by the original artist.

16 Feature Pipeline and Future Roadmap

1. Loaded PoW
2. SPHINCs Quantum Resistant Signatures
3. The Use of Steganography to Encode SPHINCs signatures as QR Codes in Images

References