Nomicle: A Self-Sovereign Identity System

1 JUN 2022

Ali Mahouk
Email: ali.mahouk@icloud.com
xTalk: mail@alimahouk

The Initiative for Novel Research & Applications in Computing

1. Synopsis

The need to identify things in a networked, digital realm is fundamental to its operation, from IP and MAC addresses at the bottom level to email addresses and domain names at the top level. In addition to coming up with a suitable identity scheme, the need to prove identity ownership at any point in time is equally necessary. Ambiguity inevitably arises when two or more entities choose the same identifier to represent themselves. Historically, trust-based centralised models have been used to solve this problem, e.g. a database on a server. The administrator of that database effectively decides on the rules of identity allocation and may choose to prevent or allow third parties to make use of their identity scheme. Users trust the admin to safeguard existing identities and abide by their proclaimed allocation rules. This leads to 'siloisation' as different parties decide they need to recreate an identity scheme that is under their control and tailored to their specific needs and rules.

In a decentralised model, trust in any form, such as timestamps, does not work for the resolution of identity ambiguity and stalemates. Instead, we propose a scheme based on consensus via cryptographic proof that can resolve identity ambiguity with the added benefit of being universally applicable.

2. On Identity

Perhaps before we even attempt to solve the problem, it would be wise to pose the question: 'what does identity even mean?' We can define 'identity' as an abstract association between two entities. In the real world, there are no absolute or fixed identities. Identity is a relative concept that exists in the minds of people, and it can only exist so long as people are aware of it and believe in it. It is a fundamental aspect of interaction. The same identity can mean different things to different people, and even those meanings can change over time, e.g. mentioning the word 'tube' in London (or even other parts of the United Kingdom) will make people think of London’s rapid transit system, the London Underground. 'Tube' in other parts of the world will most probably make people think of a pipe-like structure in a plumbing context. However, prior to 1863, the pipe-like assumption of the meaning of the word 'tube' was also the case for the people of London up until the London Underground came to exist and became popular. The identity association is one-to-many; people may associate multiple mnemonic attributes with the same entity, e.g. a book has a title, an author, a publisher, and an ISBN. A person has a name, and additionally

may have a nickname, phone number, email address, physical address, web address, favourite online alias, etc. These also apply to other entities, like businesses, and vary by country and culture.

Businesses usually allocate a portion of their budget towards the marketing and promotion of their brand alongside their products and services. Some brands provide so much value to their customers that they spread via word of mouth without having to spend money on marketing. Other brands become synonymous with the service or product they offer and enter the daily lexicon as verbs, e.g. to 'google' something, meaning to search the web for it; to 'hoover' a carpet, meaning to clean it with a vacuum cleaner, etc. The basic idea behind the concept of marketing is to expose a brand to as wide an audience as possible, repeatedly over time to the point that when people hear the brand name, that brand's business is the one that instantly pops into their minds. The larger a company's budget, the more it can spend on different forms of marketing to wider audiences. 'Google' might make millions of people around the world instantly think of the search engine with the colourful logo. However, nothing prevents someone in a small town in a more remote part of the globe from starting a grocery store and calling it 'Google'. To the local community around that store, hearing or seeing the word 'Google' will make them think of their local grocery store rather than the web search engine. The tech company can either spend more money to promote its search engine within the town to the point where people think of it rather than the grocery store when they hear or see 'Google', or the store owner could somehow come up with a huge sum of money to promote his shop around the world until he overtakes the tech company in people's minds (we are assuming that copyright and trademark laws do not exist in this thought experiment because their enforcement is based on centralisation). Marketing is a continuous process as well as a competitive one; companies need to constantly remind people of their existence and outperform the marketing of their rivals and competitors.

The second concept of relevance is a phenomenon found in myology. An athletic sprinter will naturally develop stronger and more muscular legs over the course of their competitive career as a result of the constant exercise and training they go through. We also note that the harder one trains, the more muscle mass (or endurance, etc.) one develops over time. Once they retire, should the athlete cut back on their training routine or stop training entirely, the muscles they built up will gradually atrophy. If you are no longer sprinting every other day then, as far as your body is concerned, you probably no

longer need those strong leg muscles anymore. In other words, 'what you do not use, you lose.'

Finally, we are conditioned to think that identity must be unique in order for it to make sense; that only one 'slinky' or '_phil_' may exist at a time within a particular domain. This paper proposes a new way of thinking: that nobody is truly unique; it is okay for many people to claim being someone at the same time, but what matters is who everyone else agrees to treat according to that claim at any point in time within a particular domain, from the public internet to a small home network. A digital identity system that adheres to this notion while also being based on the 'what you do not use, you lose' biological principle is one that can recycle and reuse the elements that make it work, which in turn allows it to scale efficiently without the need for centralisation.

3. Background

The world has largely come to depend on the Domain Name System (DNS) for most services that involve exchanging information over the internet. While DNS is decentralised for the most part, control over domain names and top-level domains (TLDs, e.g. .com, .org, .net, etc.) remains in the hands of a group of organisations such as ICANN and internet service providers (ISPs). Businesses and individuals have no option but to go through a domain name registrar in order to obtain an identity for their online properties, and they remain at the mercy of these organisations, who may choose to revoke access to a domain name for whatever reason. Domain names are rented, not owned. DNS also makes censorship easy. Governments can block access to information that they deem unfit for the public by simply filtering out all requests made to a particular domain name or IP address. ISPs can keep track of every domain name you access.

DNS is also being used in less-obvious ways that do not involve web pages, such as in email and mobile applications. Apps make requests and load remote resources, such as media and feeds, in the background using domain names over HTTP(S). Most email clients will also not allow a user to send en email to an address that uses an IP address directly instead of a domain name (e.g. admin@[28.92.172.57]). In other cases, the SMTP server on the recipient’s end may refuse to accept such addresses. Section 2.3.4 of RFC 2821 on the Simple Mail Transfer Protocol states that 'hosts are known by names (see ‘domain’); identifying them by numerical address is discouraged.' [1]

Let us now pose another question: why did DNS become so popular? DNS provides access to services via short, memorable names. DNS lookups are quick and the use of caching speeds up the process even further. Domain names also serve as an abstraction layer above IP addresses, allowing for changes to the latter without affecting the former. The domain name registrars safeguard each domain so that no two parties can share the same name in return for an annual fee paid by the domain tenant. For about $10 per annum, anyone can squat a good domain name and deny others from ever using it unless they pay a ridiculous price for it.

3.1. Usernames and Passwords

Any form of online registration today that involves the user creating an account requires them to create a username-password combination. Username-password logins are susceptible to brute-force attacks, aided by the fact that the username part is publicly known in some cases. Most people struggle to remember their login credentials and either write them down somewhere or pick passwords that are easy to guess. To make matters worse, people also tend to reuse the same credentials across all their accounts. This form of authentication is popular because it does not depend on external factors, i.e. all you need is your own memory to log into an account.

Single sign-on is a form of authentication that involves a trusted third party authenticating a user on your behalf. This relieves users from having to choose a password or fill signup forms, but for this added convenience, people have to pay a hidden price: their online privacy. Every time someone uses a company's single sign-on service on a website, that company is tracking the person's online activity.

3.2. Public Key Cryptography

The most significant work in the field of public key cryptography took place during the 1970s with the publishing of the RSA and Diffie-Hellman schemes. The true potential of public key cryptography was only realised after the World Wide Web and e-commerce went mainstream. In recent years, it has also become the foundation upon which cryptocurrencies came to exist. Keys come in pairs: a public key that can be widely published, and a private key that must be kept secret. Keys can be used for carrying out encryption as well as signing data (via the private key) and verifying such signatures (via the public key).

No two different private keys can ever have identical public keys. This, along with the aforementioned attributes of keys, make public key cryptography a viable foundation upon which to decentralise identity. The existing concept of a public key infrastructure requires points of centralisation, such as certificate authorities. The web of trust concept, on the other hand, is decentralised but still based on trust, nonetheless. Systems based on it, such as PGP, ultimately floundered. [2] It also does not work with arbitrary, spontaneous identifiers, and cannot be used to resolve identifier clashes efficiently.

As far as authentication is concerned, cryptographic keys are a much more secure method of doing so than username-password combinations. They do not require the user to remember any information and are significantly more resilient towards brute-force attacks. However, the key must be stored securely, and the user would have to access the key file every time they need to use it. Losing a key equates to losing access to whatever depends on it for authentication. The ability to use keys as a form of authentication depends on applications supporting it. The Nomicle system proposed in this paper is centred around cryptographic keys with the intention that it may become a ubiquitous and standardised form of user authentication that encourages more developers and software vendors to build support for key-based forms of authentication into their applications and operating systems.

3.3. Blockchain

In 2008, a seminal white paper described what was essentially a distributed database that later came to be popularly known as the blockchain. It built on previous work called Hashcash [3], and combined that with existing cryptographic protocols to create a digital ledger for recording transactions and allow for a completely decentralised electronic currency called Bitcoin that depends on cryptographic proof rather than a trusted third party. [4]

A Bitcoin transaction is a transfer of a certain amount of BTC from one Bitcoin address to another; a Bitcoin address is a representation of the public key of an asymmetric key pair. Nakamoto proposed a design that groups transactions into 'blocks', each of which includes a hash of the previous block thereby chaining all blocks together; the only exception is the genesis block which does not link to anything before it. Making any modifications to a past block would mean having to redo the proof-of-work for every single other block that proceeded that block in addition to catching up with and overtaking the

rest of the network. While not impossible, it would prove largely unfeasible for an attacker to rewrite the history of events recorded in past blocks. This unfeasibility safeguards the ledger and maintains its integrity. At any given time on the Bitcoin network, all nodes are working together towards verifying transactions and mining (a cryptocurrency term for solving the proof-of-work) the next block, thus participating in the safeguarding of the existing blockchain in the process, i.e. the nodes are selfless in that sense (but each node is incentivised by its own self-gain which comes from either transaction fees or the coinbase transaction, the financial reward that goes to the node that successfully mines the next block and serves to introduce new coins into circulation). The Bitcoin network regulates the required difficulty for the proof-of-work of the next block by setting it such that one new block is mined every ten minutes. If the hashing power of the network increases, the difficulty is increased to account for that. Similarly, the difficulty is decreased if the hashing power decreases. In 2015, a project called Ethereum was launched using its own, separate, and slightly modified version of blockchain rather than Bitcoin's. Whereas Bitcoin is only a currency, Ethereum is a platform that allows people to build decentralised applications on top of it powered by its Ether currency. Blockchain-based digital identity solutions have already been proposed and implemented on both the Bitcoin and Ethereum blockchains with varying degrees of success. As a result of being based on cryptocurrencies, they all involve some sort of financial commitment from the user because the miners that maintain the underlying blockchains are incentivised by the money they earn through transaction fees.

A digital currency, like Bitcoin, is fundamentally about solving the double-spending problem without depending on trust. This necessitates having a ledger that tracks the paper trail of money and transactions to make sure people can spend money they actually own and only spend it once. The blockchain concept is a brilliant solution to this problem. Identity, on the other hand, is a different kind of problem, with very different requirements and constraints. The primary requirement of identity is resolution, i.e. any given identifier must resolve to one and only one entity at any point in time. Resolution does not require past knowledge; the present moment is all that matters. This makes some aspects of blockchain, such as chaining, redundant, and so it does not make sense to use it for decentralising identity.

4. How It Works

A decentralised identity system requires two primary mechanisms: a means of producing identifiers and a means of distributing identifiers to those who need to use them. It should be noted that the 'users' here are in fact applications as people won't be interacting with the system directly besides entering their own chosen identifier.

A system that is complicated to set up and run is one that nobody would want to use. A primary objective of the design of Nomicle to encourage its widespread adoption is ease of use. Nomicle is a suite of two processes that run in the background on a computer. It may perhaps be bundled with the operating system and launched with other startup services.

4.1. Fortifier

An arbitrary, user-chosen identifier can be hashed ('token' is used henceforth to refer to this identifier in its hashed form), packaged into an 'identity block' along with other useful data, and then used to find proof-of-work for that block beginning at a hardcoded, baseline difficulty on the user's machine. Every time proof-of-work is found, that block is said to become 'stronger'; the system then increases the difficulty and repeats the process to solve the proof-of-work at the higher difficulty, ad infinitum. This process is called 'fortification' and is handled by the Fortifier program. An identity block is a signed, self-contained, independent entity that contains everything needed to both verify its own validity as well as to communicate with the owner of that identity, including their public key. Regardless of how many people now claim to be 'slinky' in the decentralised network, whoever happens to hold the private key of the strongest 'slinky' block will be the one that the 'slinky' token resolves to when anyone on the network wishes to communicate with them.

When starting out, the Fortifier runs at 100% intensity to get the user up and running quickly. After the first proof-of-work is found, it runs at 50% intensity with periodic, short 'sprints', quick bursts where the Fortifier runs at 100% for a few seconds. This intensity is user-configurable because hashing can be quite resource intensive when it is being carried out at a machine's full capacity.

4.2.Seeder

Once proof-of-work has been successfully found for a block, it has to now be seeded out to the entire network for everyone to use. This is handled by the Seeder program, which detects when proof-of-work has

been found by checking the last modification date of the identity block's file. Seeding happens irrespective of whether the network will accept the new block or not. A node will initially seed the new block to all its neighbours, and neighbours usually have no way of knowing where a block originated on the network unless one of them happens to somehow be aware of all network chatter to detect which node was the first to seed a new block.

Neighbours do not trust each other and will verify a new block for themselves by checking the proof-of-work. If no block for that identity currently exists on its filesystem, the neighbour saves the block to its local storage and repeats the same seeding process to all its own neighbours. If a block already exists, the neighbour will compare the solved difficulty of the new block with the existing one. If the new block is stronger, it will overwrite the existing block and the neighbour will proceed to seed the new block to all its own neighbours. If the new block is weaker, it is simply discarded and not seeded any further, and the neighbour instead sends back the stronger block to make the node aware of the fact that a stronger version of the identity exists. The node can verify that fact for itself and replace its weaker copy with the stronger one. A node will not overwrite a weaker block with a stronger one in the case where the weaker block belongs to its local machine (i.e. it holds the private key of the public key inside the block and the block's token matches the identity currently set for that machine). This is to allow machines to continue fortifying their weaker version of an identity in hopes of eventually obtaining the strongest version.

Nomicle currently employs a broadcast model to seed blocks. A new block would not take long to propagate across the entire network, assuming it is valid and the strongest version of its token. This propagation can be thought of as an infection and the SI epidemic model can be applied to validate this claim [5]. Nodes that have yet to receive the new block are susceptible; nodes that have already received it are infected. The growth rate of the number of infected nodes becomes exponential until the network is saturated with the new identity block.

An identity block is around 249 bytes in size (or 248 bytes depending on the signature size) and fits in a single UDP packet. 1 gigabyte of memory can store about 4,016,064 identities.

4.2.1.Overtaking

Nomicle incorporates a mechanism called overtaking to make things fairer for those who have invested a lot of time and power into their identities. It works by introducing a threshold that a difficulty needs to be compared against before deciding on the dominant specimen of an identity. The threshold is simply a percentage value that represents how much higher a difficulty of a new proof-of-work needs to be relative to the older difficulty. The percentage value is defined within the Nomicle protocol specification, meaning that nodes can know which threshold to compare against based on the protocol version a particular identity block is using. Nodes will always prefer a lower threshold satisfaction with a newer protocol version over a higher threshold satisfaction with an older protocol version. This also protects against attackers intentionally using an older protocol to force a downgraded check against what could potentially be a lower threshold from a previous protocol version. It also serves the purpose of recycling identities; e.g. an identity that has been fortified to a very high difficulty but has since been abandoned for a long time will not be updated with the latest protocol version. Anyone trying to use that identity now will be using the latest protocol and will hence be able to claim it more easily if the threshold in the latest specification is lower.

The following example will hopefully make things easier to understand. Let us say that a certain node receives a new identity block for 'foobar' with a proof-of-work solved at a difficulty of 10. The same node now receives a block for 'foobar' signed by a different key and a proof-of-work solved at a difficulty of 11. Assuming the current protocol specification states that the overtaking threshold is 20%. The node calculates 20% of the difficulty of the previous 'foobar' block and finds that the threshold is currently 12 (20% of the previous block’s difficulty of 10, which is 2, summed with its current difficulty of 10). The new block’s difficulty of 11 is below that threshold, therefore the node will reject it. Now let us say that some time has passed and the owner of the new block managed to solve an even more difficult proof-of-work at a difficulty of 13 while the previous block remains unchanged at 10. The node calculates the threshold again and compares 12 to the new difficulty of 13. The new block has now overtaken the previous block; the node overwrites the previous block with the newer one, and the owner of this block is now the dominant specimen. The new block similarly and gradually spreads across the network until it is dominant on all nodes.

Because the overtaking threshold is a percentage value rather than a fixed one, it makes it harder to overtake powerful identities that have had a lot of time and/or power invested into them. The threshold serves as a delay to give the current dominant specimen a chance to catch up to the new competitor and maintain their lead without being unfair to either party. If they cannot keep up, the competitor fairly claims the dominant position. Returning to the previous example, had the owner of the previous block solved a proof-of-work for a higher difficulty in the meantime, the threshold would become a higher value, and the newer block would not have overtaken it at a difficulty of 13. The owner of the newer block would need to go further and solve an even more difficult proof-of-work. This game of catching up can go on indefinitely until the owner of the newer block gets lucky and finds multiple proofs-of-work faster than the owner of the previous block. A good metaphor would be a car trying to overtake another moving car on the road. The overtaker must not only match the other car's speed at any moment in time, but must also accelerate faster in order to be able to get in front of it.

4.2.1.Trust Authority

Nomicles include an optional Trust Authority field. A trust authority will typically be an SSL/TLS-enabled domain name. The domain owner may use their TLS private key to sign a nomicle (this step is done prior to the identity owner signing the nomicle). Nodes can then fetch the trust authority's digital certificate and use its public key to verify the authority's signature inside the nomicle. This adds an extra layer of authentication in cases where the identities themselves can be arbitrary but ensuring that they come from a particular domain is necessary (e.g. an organisation).

4.3.Using Nomicle

The Nomicle programs were designed to run standalone with shared files. It is possible to run a setup that is purely for fortification or seeding. This paper assumes a typical setup, which does both functions. When starting up Nomicle for the first time, you have the option of passing an arbitrary string as an argument to the Fortifier to use as your personal identifier. There are no length or format restrictions on the chosen string although depending on what you intend to use your identity for, you may want to take certain constraints into account, e.g. if you intend to use your identifier to host a website, it must be URL-friendly. Note that Nomicle identifiers are case-insensitive, so 'wolf', 'Wolf', and 'WOLF' are identical. This string

will be saved to the identity file located at /usr/local/share/ncle/id on Unix-like systems and at %APPDATA%\NCLE\id on Windows (note that the filename lacks a file extension by design, but it is in fact simply a text file). You may edit the identity file using any text editor at any time to change your identifier; just be sure to restart the Nomicle programs after you save your edit. Make sure your text editor does not automatically append a .txt file extension to the filename. Running a pure seeding installation does not require an identifier to be present in the identity file.

The Seeder maintains a block repository on the local filesystem. By default, blocks are stored at /usr/local/var/ncle/blocks/ on Unix-like systems and at %APPDATA%\NCLE\blocks\ on Windows. A block's token, which is a SHA-256 digest, is used as its filename followed by the .ncle file extension, e.g. the block for the identifier 'foo' would be named 2c26b46b68ffc68ff99b453c1d30413413422d706483bfa0f98a5e886266e7ae.ncle. The block repository path, among other options, is user-configurable. You could, for instance, point the path to a directory somewhere inside your web server's root directory and make your Nomicle repository available to applications over HTTP(S).

Applications can simply open and read the block files of the identity they need. While the aforementioned file naming scheme is the convention, applications should not trust that the token inside the file matches its filename as the filename may have been altered somehow. Apps can parse a block file themselves by following the Nomicle identity block file specification or by using one of the available Nomicle libraries. In a nutshell, the service an app can expect from using the system is an elliptic curve public key that is guaranteed to be tied to a token it wishes to interact with, assuming that token has been claimed by someone on the network, along with a private key tied to the identity of the machine it is running on, which would be located in PEM format at /usr/local/etc/ncle/privkey.pem on Unix-like systems and at %APPDATA%\NCLE\privkey.pem on Windows. Working with cryptographic keys will require a library such as OpenSSL. Consult the documentation of the specific programming language you are using. With the local machine's private key and someone else's public key, it becomes possible to carry out encryption using the Diffie-Hellman cryptographic scheme, to sign data, and to verify signatures in a completely decentralised manner.

4.3.1.Probing

Inevitably, an application will require the block of an identity that does not exist on the local filesystem. For this, the Seeder maintains a probes file, which is a plain text file containing a list of identities applications need in the repository. If an application requires the block for 'slinky' and discovers that the 05a1dfc495bf88277862a8a97c7834a1f15c5ab926b16599d2b94d93948e7ae5.ncle file does not exist, it may open the probes file (/usr/local/var/ncle/probes on Unix-like systems or %APPDATA%\NCLE\probes.txt on Windows) and append a SHA-256 hash of 'slinky' (05a1dfc495bf88277862a8a97c7834a1f15c5ab926b16599d2b94d93948e7ae5) followed by a newline to the end of the file. The Seeder has no means of notifying apps when a block they have been probing for is added to the repository, therefore apps must periodically poll the repository to know once the file exists. The Seeder automatically removes the entry for an identity from the probes file once its block has been found.

4.3.2.Hosts

The Seeder maintains a list of known Nomicle installations in the hosts file (/usr/local/var/ncle/hosts on Unix-like systems or %APPDATA%\NCLE\hosts.txt on Windows). A user may manually modify this file to add or remove IP addresses and port numbers as they wish. The Seeder uses the broadcast address to discover other installations on its LAN.

4.4.Configuration

Configuration files consist of key-value pairs delimited by newlines. A graphical interface may be created to allow the user to configure their installation without having to delve into the configuration files themselves. Applications may also parse these files to determine where the user's private key and identity repository are located on their system. The following aspects may eventually be configurable by the user:

* Port numbers: these would be standardised so there needs to be a good reason to use anything other than the defaults. This might be the case in a custom Nomicle setup running on an intranet.
* Whether to use the local system username (the value of the $USERNAME variable on Unix-like systems or %USERNAME% on Windows) as a default identity in case the identity file is blank.
* What directory to use as the identity block repository.
* The path to the user's private key.

* Blob mode: whether to use a custom file as an identity rather than the contents of the identity file, in which case Nomicle uses a hash of that entire file as-is. This means a user might set something like a photograph to represent their identity.
* Black/whitelist identities: this may be done on a token, public key, and/or trust authority basis. The Seeder can check this list and accordingly accept or discard certain identities.
* Fortification intensity: this is a decimal value between 0 and 1. On older or more limited hardware, setting this to a lower value will help avoid degrading system performance and battery life. Powerful or dedicated machines may want to set this value to 1 to take full advantage of their hardware.

5. Security Implications

Two of Nomicle's primary intended uses are the encryption and signing of messages. While an encrypted message is in transit, it is possible for the owner of the sender's identity to lose ownership of that identity due to someone else overtaking them. In such a scenario, the contents of the message will remain safe and nobody but the recipient will be able to see them. Each message can include the public key of the private key that was used to encrypt it. So while this public key would no longer match the one contained in the corresponding identity block (because the key in that block is now the one belonging to the new identity owner), the recipient would still be able to decrypt the message. They would not be able to trust that message, however, unless they had previously dealt with the sender to compare the key embedded in the message with an existing copy of the key they had retained from a previous message when the sender was still the owner of that identity. Nomicle itself does not attempt to provide resolution in such a case.

It goes without saying that the security provided by the whole system rests on the security of a user's private key, a responsibility that ultimately rests upon the user themselves.

5.1.Denial of Service Attacks

An attacker may attempt to flood the network with bogus or weak blocks of common identity tokens that are unlikely to be accepted by other nodes. Such an attack would not compromise the integrity of the system but it may cause disruption by hogging system and network resources. It would, however, be confined to the attacker's immediate neighbours. The number of potential neighbours is determined by the

attacker’s hosts file, which may have grown organically by normal operation of the Seeder daemon or artificially through the attacker's own compilation of known Nomicle hosts.

6. The Future

With the way affairs are currently going, the future is looking incredibly centralised, much to the benefit of a handful of large corporations and the chagrin of users. To enable the unshackled growth of the next generation of technologies, the world needs to be more open and decentralised, much like the nascent days of the internet and the myriad technologies that both gave rise to it and that it enabled, like TCP/IP and the World Wide Web.

6.1.Usage Examples

6.1.1.IP Address Exchange

Two people wish to exchange a large file. They do not know each other’s IP addresses. However, they do know the alias by which each likes to go on the online forum they frequent. Either (or both) of them publishes a message to the other that contains their own IP address. As soon as one of them receives the other’s address, they input it in into their file transfer application. The application connects directly to the address and uses its own protocol to send the file to the person on the other end. This example is not necessarily referring to the File Transfer Protocol; it may very well be for requesting a web page from a web server to a browser without relying on DNS.

6.1.2.Anonymous Publishing

A human rights activist would like to publish a video without associating their real-world identity with the video file from fear of retaliation. They also want to somehow be reachable by members of the press and the public. The activist computes a hash of the video file, sets it as an identity, and fortifies it for a few days (the chances of identity clash would be negligible anyway on account of the hash being unique to this file). They then publish the video on the internet. Anyone who would like to contact the anonymous publisher can compute the same hash of the video file, write their message, and address it to the hash. The activist can then receive these messages and reply back without ever revealing who they are or that they are even associated with the video and their responses can be verified by anyone.

6.1.3.Consensus-Based Message Routing

An alternative to Nomicle's current broadcast model of distributing data is consensus-based routing. As the solved difficulty of each identity is quantifiable, any two difficulties can be compared to one another. We can use this property to apply a leader-selection algorithm. A node that has the most powerful identity in a group is designated as the leader of that group and acts as their gateway. Rather than broadcasting messages, nodes talk to their gateway to help them route messages to their destination as well as to probe for identities. This approach generates less network traffic.

Consensus-based routing is a topic for its own paper.

6.1.4.AI and Agents

This example is a hypothetical but somewhat plausible one. In the near-future, AI agents could take on the form of portable, polymorphic binaries. The binary file would represent, if you will, their existence and self-consciousness, and the agent is capable of modifying its own binary to incorporate new elements as it learns and becomes more advanced as opposed to the traditional binaries found today where new information is always stored in external files created by the binary. Agents might traverse your computer's filesystem to browse files and carry out operations. As much as the age of the web and cloud computing helped break down the physical boundaries between geographically separated machines, this would take it a step further where the boundaries between information on people's hard drives around the world gradually disappear. Agents would travel between machines on various missions (either self-directed or human-directed) to gather information or carry out tasks. Specialised programs akin to our own airports would act as bases on every machine where agents land or leave that machine over the network, subject to rules much like our own immigration laws that dictate which agents may go where on the local filesystem, how long they may remain on the system, and what kind of operations they are allowed to carry out. In such a case, each agent would need to have its own identity that uniquely represents it incorporated into itself; this would act like its passport. An open, decentralised system like Nomicle would scale well in such a scenario, allowing new identities to be created on-the-fly as new agents come into existence, and for every identity to be easily checked without depending on a centralised model that would inevitably grind to a halt past a certain scale. In the real world, an airport is limited to the number of passengers it can handle simultaneously at a given time by its physical capacity, but in the

digital realm, no such limits exist so long as more servers are readily available to handle the increasing load. As our knowledge on how to create new agents faster and more efficiently (e.g. some sort of templating system) improves, a single computer might be able to generate thousands of new agents within minutes. With computers becoming more powerful and increasing bandwidth limits becoming cheaper, a point may arrive where hundreds of millions of agents (or perhaps even billions) may be arriving or departing simultaneously on a given network.

6.2.Fortification as a Service

Based on patterns that emerged around new technologies in the past, it would not be unreasonable to predict the emergence of businesses that specialise in identity fortification and offer that as a service to other businesses and individuals. People with an interest in using ubiquitous tokens as their identity are likely to face stiff competition from others who also want to claim that identity, whether for good or ill, as will major brands. Anyone relying on an average laptop will be no match against someone else using an application-specific integrated circuit (ASIC) device specialised in hashing that is capable of trying out different hashes at a much faster rate. It would be the equivalent of two people attempting to break down a wall; one using a hammer and the other using a wrecking ball. Fortification as a Service (FaaS) providers would employ entire farms of specialised hardware to help maintain the mission-critical identities of others. This will inevitably require some degree of trust in the service providers as they will have access to the private key of each identity. However, it is also possible to separate the fortifiers from the signatories by having the fortifier submit a block that was successfully mined to the signatory for them to sign it with their private key and send it back to the fortifier for seeding to the network. The signatory in this case can be the identity owner themselves. Major companies and governments that can afford the budget might choose to host their own farms rather than outsource the task. More obscure identifiers may not need FaaS as they are likely to have little or no competition, and any competition would more probably be employing general-purpose hardware.

6.3.Mobile Devices

Regardless of how powerful mobile devices (with the exception of laptops) currently are or will become in the future, their mobile nature makes them unfavourable for identity fortification primarily

because the constant process would tax the device's battery. People use mobile devices to come online sporadically throughout the day for short periods of time; network connections are short-lived and alternate between using Wi-Fi and cellular networks. This is not to say that users cannot use their existing identities on their mobile devices. Perhaps operating system vendors might one day build a secure location within their OSs for housing identities and providing applications with access to them in a secure manner. The OS would use some sort of cloud service to make sure a user always has the strongest version of their identity synced with their device while the fortification itself would take place elsewhere.

7. Nomenclature

Nomicle means 'named key' and is derived from the French words nom ('name') and clé ('key'). In English, the -cle in Nomicle is pronounced like you would pronounce it in 'particle'.

It may be used as a proper noun to refer to the entire system or as a common noun to refer to the individual identity blocks.

8. Conclusion

In an increasingly digital world, identity plays a critical social, economical, and geopolitical role. It is a complex problem that has no successful, decentralised solution thus far. Through the pairing of cryptographic keys with mnemonic identifiers and combining them with proof-of-work fortification, we have proposed a universal, scalable, decentralised solution called Nomicle that is based on cryptographic proof rather than trust and allows us to send messages to specific recipients across a network without the need for a central server or knowing their IP address. By quantifying the abstract concept of identity, it becomes possible to decouple it from data and build decentralised services where ties can be broken and and conflicts resolved.

References

[1] Internet Engineering Task Force (IETF), 'Simple Mail Transfer Protocol,' https://www.ietf.org/rfc/rfc2821.txt, 2001.

[2] H. Story, 'Why did the PGP Web of Trust fail?' https://medium.com/@bblfish/what-are-the-failings-of-pgp-web-of-trust-958e1f62e5b7, 2018.

[3] S. Nakamoto, 'Bitcoin: A Peer-to-Peer Electronic Cash System,' https://bitcoin.org/bitcoin.pdf, 2008.

[4] A. Back, 'Hashcash - a denial of service counter-measure,' http://www.hashcash.org/papers/hashcash.pdf, 2002.

[5] M. Newman, In 'Networks,' 2nd ed., page 612, 2018.