dappy

A new protocol, network architecture and resolution paradigm for web PKI and domain names. Powered by blockchain, engineered for the next generation of secure web services.

Raphaël Fabre, dappy whitepaper v2, december 2022

This paper presents the general architecture and motivations for dappy, an alternative protocol to both the Domain Name System and the web PKI in its current form (based on Certificate Authorities). Dappy's design is inspired by dane standard and blockchain technologies, it relies on the combination of a leaderless, distributed state replication system, and co-resolution. Dappy provides a resilient, trustless naming and public key infrastructure, its superior security and privacy-preserving settings have the potential to avoid dozens of cyberattack or data leaks to existing web services, in addition to allowing the next generations of privacy-focused, decentralized, and disruptive public web services to exist.

Importantly, dappy domains can be integrated easily, they do not collide with any existing name system as domains will be scoped under a new TLD.

This paper is the second whitepaper (v2), it includes changes and revisions over the previous version, which are based on many discussions the team had the opportunity to have with security researchers and engineers during years 2021 and 2022.

Motivations

The Domain Name System, and Certificate Authorities, responsible respectively for naming and the distribution of TLS/HTTPS certificates (web PKI) power millions of enterprise web portals, social medias, blogging platforms, and government digital activities today.

We witnessed the last ten to twenty years the institutional push against abusive usage of personal information by giant internet companies and in favor of a more decentralized and transparent internet governance, and in parallel the constant growth of SaaS, enterprise and retail web services, in number, but also a growth in terms of the criticality of the data, and operations that are performed.

Some pure players companies rely exclusively on their internet applications to sell goods or services, other companies in traditional industries

continuously expand their digital activities in order to remain competitive, turning into online activities, fields that were operated outside of the internet and even computers. Some examples of very critical web services (B2B or even B2C) are healthcare, Insurance, supply chain, manufacturing, banking, governmental institutions, cryptocurrency or securities and commodities trading.

The web is growing fast, unfortunately, as web portals carry more and more critical data and value, they become more and more appealing to hackers, or malicious organizations, motivated by different things.

For many researchers, engineers and observers, it is in 2022 obvious that the two main components responsible for web identities (DNS and the public CA based web PKI) are limited when it comes to providing the privacy, censorship resistance and agility levels required by the public web platforms of the 21st century.

Cybersecurity of public web service is becoming a real issue, primarily because of the huge amount of money that is lost in hacks and data breaches.

Incidentally, a new field is growing and expectations that people put in this technology too. This field is web3, and technologies or digital products related to blockchain, digital tokens and cryptocurrencies. Dapps (Decentralized applications) are at their infancy stage, in addition to security they seek maximum levels of decentralization, openness, transparency and trustlessness. NFT or token-based games, interfaces for managing DAO's (Decentralized Autonomous Organizations) and Decentralized Exchanges (DeFi) are examples.

Many web services do need more trustworthy protocols and infrastructure, and we also think that many others (especially DAOs and web3 related services) cannot exist because of the centralized structures, or security flaws of the DNS and CA based web PKI. Dappy aims at providing a more **efficient, transparent, decentralized, secure and resilient** protocol for identifying resources on the internet network (name system or naming system), and allowing clients to communicate with those services in the

most private way possible (web PKI). **Except for co-resolution, dappy only articulates, or assembles existing concepts and technologies.**

Dappy is different from the two systems it aims to partially or completely replace, but should have a similar governance structure as the Domain Name and web PKI (Public Key Infrastructure) systems, dappy will exist and evolve in a consortium fashion.

The CA based web PKI and the DNS both become less secure as companies or organizations join, dappy becomes more secure as companies or organizations join the network and is therefore capable of providing superior privacy and security levels by many orders of magnitude. This is precisely because dappy applies a distributed trust paradigm for both storage and resolution (service discovery).

After the prerequisites, this paper goes through two main sections, first the problems and limitations of current systems, and second the solution that is proposed. Informed researchers may go directly to the Dappy, web PKI and domain names, the new way section.

1 - Prerequisites

Internet Protocol

IP (Internet Protocol) is the protocol essentially responsible for routing internet packets from a source to a destination, solely based on IP addresses. Thousands of internet service providers and millions of routers implement those protocols and power the internet. BGP protocol is another low level protocol used for exchanging routing information.

The Domain Name System (DNS)

The DNS is the main naming protocol for the internet. It is deployed above IP to allow people, companies and governments to expose web services through domains that humans can easily remember like *example.com* . DNS is hierarchical (or pyramidal), at the top is ICANN, a non-profit organization

that allocates the TLDs to various companies or countries (AFNIC manages .fr, Verisign manages .com etc.)

Web PKI (or the Certificate Authorities system)

IP (and BGP), and the DNS are considered insecure, because of the low level of authentication and possibilities of spoofing and man-in-the-middle attacks. Put simply, we are never sure of who we are talking to (*12.12.12.12* can pretend to be *13.13.13.13*), thus we're never sure of the accuracy of the responses, and data received.

The web PKI or Certificate Authorities system is the true identity guardian of the web today (2022). Certificate verification is the operation that gives a definitive yes or no for whether or not a secure connection (TLS) can go through. We are referring here to HTTP+TLS or anything+TLS connections.

Note: when we use the term *Web PKI*, we will mainly refer to the main public web PKI in its current form, based on Certificate Authorities.

Note: privacy in the context of the web can mean two things, the first meaning is that a user is completely anonymous when he browses the web, VPNs or mix networks for example are technologies that anonymize traffic and allow this first kind of privacy. The second meaning is that a client properly identifies a web server (eventually through a domain name), does proper key exchange with it (first part of TLS) and is able to communicate in a truly end-to-end encrypted fashion, without any possibility of man-in-the-middle interception. Dappy's new approach to service discovery and domain management provides privacy of the second type.

How does web PKI work ?

A PKI is a program or protocol that assigns cryptographic public keys to computers in a network. It often includes a list of certificate authorities (sometimes called trust anchors) that are capable of issuing signed certificates, and a cryptographic verification protocol. A certificate authority is identified by a cryptographic public key plus common name and other data embedded into a root certificate. Different browser vendors, companies or operating systems may have a different list of certificate authorities. In this

paper we focus on the concept of public PKI, typically used for the public web.

A browser for example will have 168 certificate authorities referenced in its codebase. Each time a server is reached for HTTP+TLS connection, it has to present a certificate that will be checked against the list of root certificates.

2 - Problems and limitations

The three limitations and security flaws that are presented constitute, or create attack vectors that can be exploited by attackers. Attackers will typically try to take down a web service, or usurp it so that they can extract whatever value or data from users, spy on, intercept, block or spoof client-server communications.

Limitation and security flaw number one : web PKI

The verification operated by a browser (or in some cases by the operating system) to eventually authenticate a server consists in finding one certificate authority that signed the certificate presented by the server.

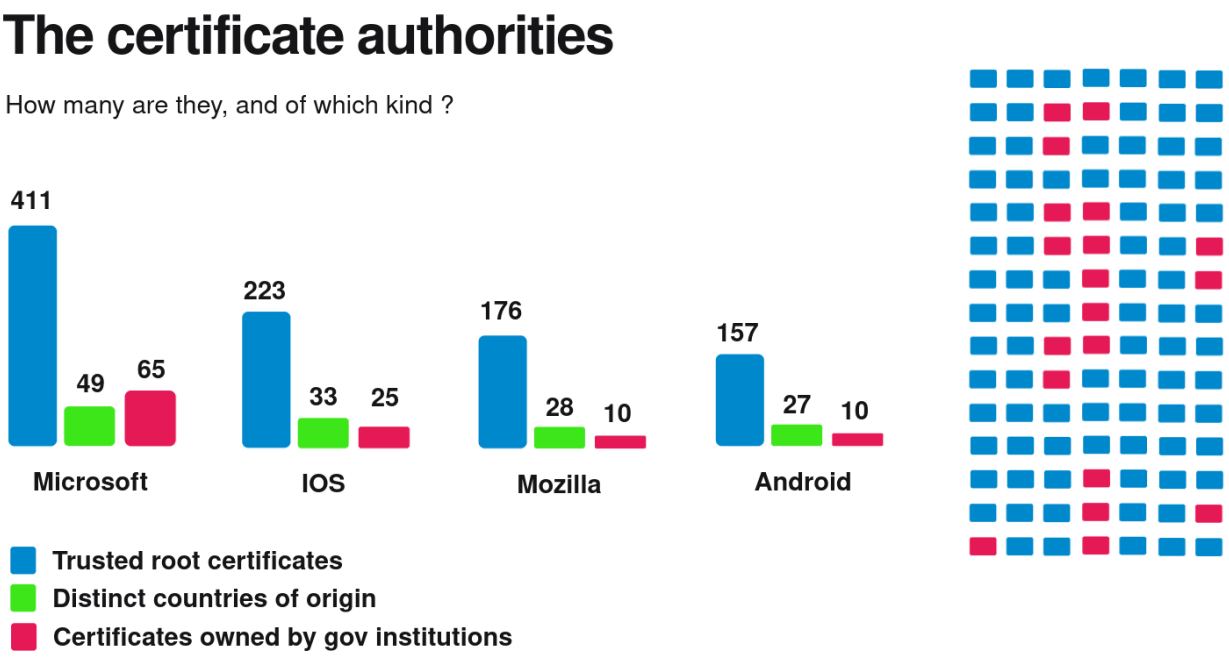
Hundreds of private organizations and government institutions can provide counterfeit (or fake) certificates for all public domain names, **trust is therefore not distributed but oddly concentrated in a large number of actors**.

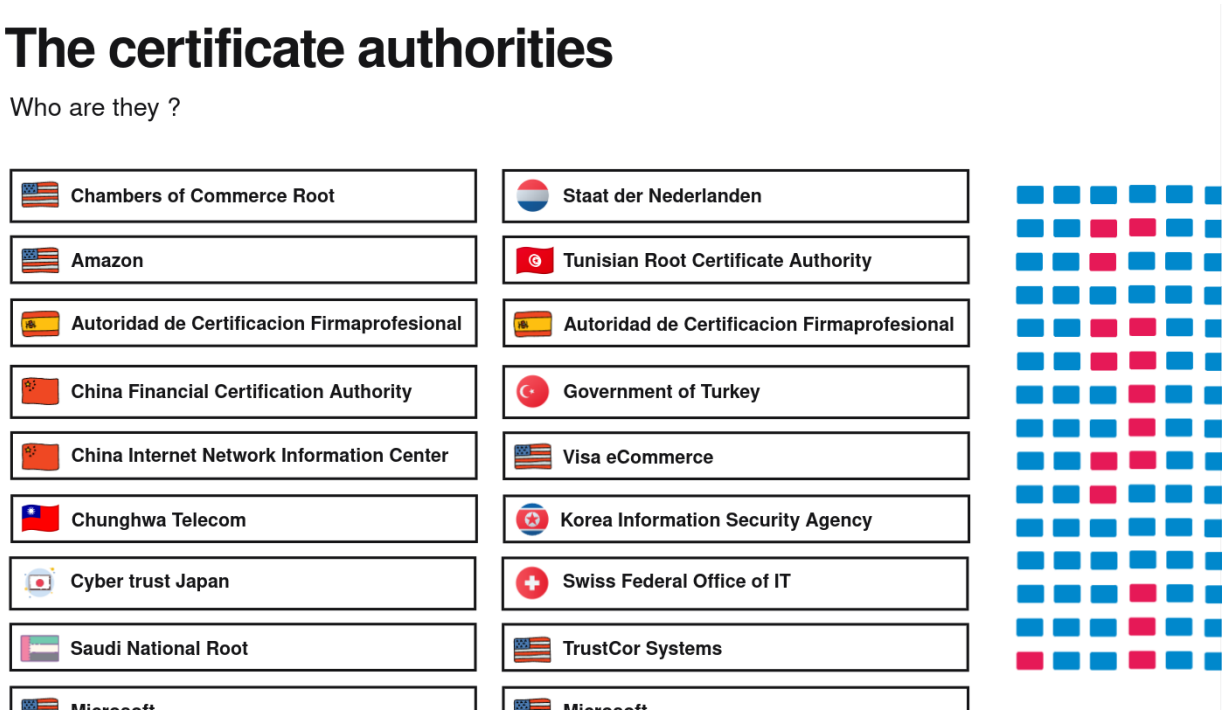
As an example Amazon and Chunghwa Telecom both have an entry in Mozilla's CA Certificate Program (among 166 others) which is probably the most conservative and secure web PKI program. One of them may conduct Man-In-The-Middle attacks globally or in some specific region, and bypass the security and privacy supposedly offered by the CA system (or HTTPS protocol).

(1) “the security of your site is only as strong as the weakest CA that clients trust”

(2) “The public CA model upon which TLS has depended is fundamentally vulnerable because it allows any of these CAs to issue a certificate for any domain name.”

(3) "[SSL has an] obscure distribution of trust."





Note: this listing of Certificate Authorities operated by governments, or private companies is not meant in any pejorative way.

Open CT

Open Certificate Transparency (Open CT) (6) is an Internet security standard that aims at making certificate issuance publicly auditable. Open CT does prevent misbehavior by CAs in some cases. We argue that this technology makes the overall trust structure of the web again more complicated, removes some vectors of attack, and adds some others.

Open CT was set up as a security layer over the web PKI, that was set up as a security layer over the insecure DNS. Are we stuck in a never ending loop ? Is it good that the security stack of the internet goes up forever ? Soon there will be a new protocol or standard responsible for checking the logs or Merkle trees of open CT companies.

All the technologies that are referred to here (DNS, web PKI, Open CT) use the same paradigm of trust being put in a company or private service, that is why new protocols, security verifications are constantly added onto the security stack of the web. We will see that dappy's distributed trust approach removes this need almost entirely, as multi-party validation or endorsement is inherent to the system.

Limitation and security flaw number two : domain verification

Let's consider that the companies of the web PKI do not act maliciously, show no misbehavior and are not hacked, they can still fail to properly verify a certificate request. Mismatching a certificate signing request (CSR) with the legitimate owner of a domain name leads to a fully valid certificate being issued to someone else than the legitimate owner of a domain. Of course from here Man-in-the-Middle and data leaks can realistically happen.

Difficulty to perform domain verification, and to a greater extent to automate it is due to the poor security of the IP and BGP protocols, CA will use insecure DNS and HTTP to authenticate a web server before signing a certificate.

(4) *"the purpose of HTTPS is to protect you against an attacker who controls the network, but this type of domain verification is completely at the mercy of an attacker who controls the network."*.

Limitation and security flaw number three : DNS

Three distinct actors are responsible for the management and storage of domain names, and data exposed through DNS (TXT, CNAME records etc.), ICANN, Registries and Registrars.

ICANN assigns one registry per TLD (*.com*, *.net*, *.fr* etc.), and maintains the root zone which includes the IP addresses for the TLD registries. We can say that it is unlikely that ICANN goes through a major hack or downtime given the criticality of its mission, and the high replication of the root zone across many countries.

Some critics may point out that this non-profit organization is in the end controlled by the United States, or acts in a rather authoritative way. We could agree on some of those points, but this paper wants to give a pragmatic and technical analysis of the existing protocols, and potential solutions for the identification and security of internet resources, therefore we will not extend on political or philosophical arguments.

Registrars and registries exist by the hundreds, they can more realistically be hacked, and were precisely hacked in the past.

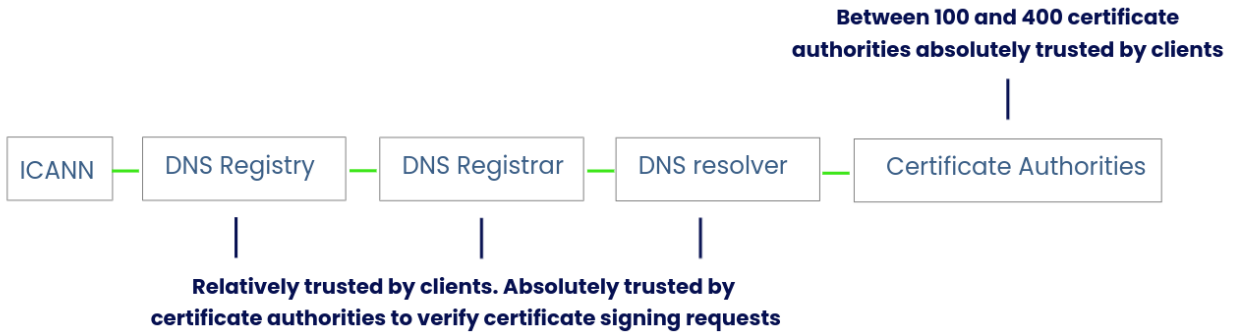
A hacker may manage to log in a registrar website on behalf of the client, or even access the back office for employees or penetrate the enterprise network. From there he can simply change the DNS data, point the domain to his servers, and in a few minutes get a brand new HTTPS certificate. From there the entire chain is in control of a malicious agent, with a lookalike website he can exfiltrate any kind of data. This happened to a brazilian bank in 2017, it lost control of the domain for 8 hours. (7)

The data managed by registrars flows to the registry, which is massively queried by DNS resolvers around the world. Of course, hacking or misbehavior of TLD registries can happen as well, with the same consequences, only at a bigger scale.

As an example, a security consulting company Palisade showed in 2021 that (5) all the domains under *.to* (Tonga's national registry) could be potentially controlled by a hacker, with only a SQL injection attack.

In relation with the two previous issues about trusting the web PKI and domain verification. Let's say the web PKI works great, certificates are properly issued to domain owners and organizations within the web PKI will not misbehave. The companies composing the DNS are trusted centralized entities, they come in addition to web PKI, they can't be bypassed if one wants to expose an internet resource through a domain name. They constitute a whole separate attack surface that can condemn critical websites or APIs, lead to data leaks and espionage.

The trust model of the DNS and web PKI based on certificate authorities



Industry-wide need for a better system

The Domain Name System and the web PKI should not be two different networks, we argue that the CA system is a superfluous subscription service. Certificate Authorities rely on the DNS to authenticate a server and issue a certificate to the legitimate person or organization, it means that DNS is in the end the master identity provider (you have to buy a domain first, and then eventually ask for a certificate, it does not work the other way). **This odd two networks structure is the reason why certificate renewal exists in addition to domain name renewal.** The constant shrinking (currently three months for Let’s Encrypt certificates) of the validity period is caused by this,

CAs need to be sure the legitimate owner is in possession of a signed certificate that it endorses. Of course Open CT is not needed if the CA system (as a separate network from DNS) disappears.

TLS certificates management should be directly handled in the name system, this way the link between a TLS certificate and a domain is as obvious and simple as the link between a TXT record and a A record in the name system.

DANE is great

DANE is a protocol that precisely aims at making the name system responsible or partly responsible for the TLS identities, with a new record type TLSA, in addition to TXT, A etc. We invite you to eventually go through the RFC.

(2) *“Certificate usage 2 is used to specify a certificate, or the public key of such a certificate, that MUST be used as the trust anchor when validating the end entity certificate given by the server in TLS.”*

DANE proposes multiple configurations, Certificate usage 2 allows a domain name owner to directly control the trust anchor that will be used by browsers, and bypass the certificate authorities. Many challenges and momentum issues prevented DANE from being widely adopted.

The first is the lack of encryption and server authentication in the DNS protocol, effectively preventing any personal computer or browser from ever getting a trustworthy response. DNSSEC could be implemented but it often stops at DNS resolver level, the browser being left with an untrustworthy TLS certificate response from the DNS resolver.

Secondly, security and TLS certificate management of DNS domain names had already massively moved to the web PKI or certificate authorities system. Which for years have been providing “secure enough” certificate verification services.

Thirdly, the DNS - a pyramidal and centralized network - becoming responsible for all things web security raises political or governance

concerns. Web PKIs have a more horizontal, consortium-like structure, it is seen by many as a good complementary brick, spreading the responsibilities and power over internet identities onto two different structures.

DNS and the web PKI become less secure as they grow

Let's summarize. Security of web services is limited, because of its two-networks structure that adds friction and enlarges the attack surface. Lot of tension is building around certificate authorities, who should be responsible for the web PKI ? who should join or leave ? based on which criteria ?

Each new certificate authority constitutes a new single point of failure. In the DNS as well, a new TLD or a new registry constitutes in some way a new single point of failure. **We argue that the DNS + CA based web PKI systems become less secure as they grow**, and are not capable of scaling from a security perspective, preventing public web services from ever reaching decent levels of privacy and security.

3 - Dappy, web PKI and domain names, the new way

We propose a new name system, TLS identities management solution, and service discovery mechanism. **At the opposite of the DNS + web PKI systems, dappy distributes trust anchors dynamically, it becomes more secure as companies join because it uses a distributed trust mechanism.**

Three aspects are key to understanding dappy.

Leaderless state replication system

The storage and management of the domain names and TLS identities have to migrate to a **blockchain, or blockchain-like leaderless state replication system**, members must be independent (in the same way that members of a web PKI or a blockchain are independent), have a copy of the ledger, and

only trust cryptographic signatures for state changes. Now there is a relatively decentralized network, they can still take collective decisions but there is no center of decision that can be corrupted and lead to spreading of counterfeit data to the whole network. Users can manage a domain sovereignly with a private key, domains could even belong to DAOs and be updated only based on agreement between 5, 10 or 20 co-owners.

We call this set of nodes connected to the same state replication system and maintained by independent organizations a dappy network.

Note: A smart contract technology like Ethereum can be used, or if one values more parallel computations and tokenless/gasless approach (which Ethereum and PoS platforms do not provide), major database technologies already offer 80% of the implementation. What matters is that users have direct ownership of their domains and assets : state changes must be governed by cryptographic signatures.
Note: how the domains should be purchased or renewed is discussed later.

Point of view matters

Putting a name system on a decentralized blockchain is not enough. Domestic computers and smartphones, which easily represent 99.9% of the devices that will interact with blockchain are left aside from the blockchain consensus, they have no realistic way of taking part, or even verifying state changes, or observing the full state of any layer 1 blockchain or other decentralized ledger, we see no reason why this would change in the next 5 to 10 years at least, the [light client - full node server] model will remain.

What we mean is that although blockchain platforms may be decentralized, web services that need to connect to them rely on unique endpoints, or trusted centralized API services. Users may have true ownership of their NFTs or cryptocurrency, but in many cases have no way to confidently read the blockchain state (as light clients).

Understanding the different point of view a light client, and a full node have is very important for understanding why we propose both tokenized blockchain and no-token permissioned ledger as possible solutions for the storage and management of domains. It may be that securing the

accessibility by light clients is the most important thing to do, and if done properly, a 20 members permissioned network may provide 100x censorship resistant level than a permissionless PoS platform with centralized point of access for light clients.

The capability for light clients to read securely in the ledger and without any single point of attack or trust represents at least 50% the challenge, which we precisely tackle with co-resolution.

How are domains stored ?

In the DNS, we typically go through a registrar to manage a domain name, in reality we manage a zone, and not a single domain. If one purchased *example.com*, he or she can manage a DNS zone, add TXT, CNAME, NS records, and also subdomains with the A and AAAA records. Management of a zone in dappy is the same. Since dappy imposes less intermediaries and allows direct ownership (no login, no email), a user should also be able to manage a zone with a JSON file and command line interface that signs zones, and send signed messages to the dappy network.

Once any network member receives a signed zone, it can update its local state, and gossip the message to other members, that is how updates are propagated (different blockchain or DLT protocols may do it a different way).

Co-resolution

The resolution, or service discovery mechanism on the client side must change as well, a unique resolver on the network (like a DNS resolver) whoever operates it would break the trustless properties.

Many nodes must be reached directly by the client (browser, operating system or arbitrary program) to resolve IP addresses, TLS certificates and even other resources that are linked to domains and subdomains (a zone in the DNS or dappy name system can contain all sorts of data). **Co-resolution allows clients to not trust anyone in particular, but again, to spread the trust over the members of the dappy network that are relatively trusted.**

After 5, 10 responses have been received from members of the network, the client can do consensus on the answers, check for full homogeneity or

relative homogeneity, and choose to go forward or display an error message based on hard-coded or manually configured synchrony constraints.

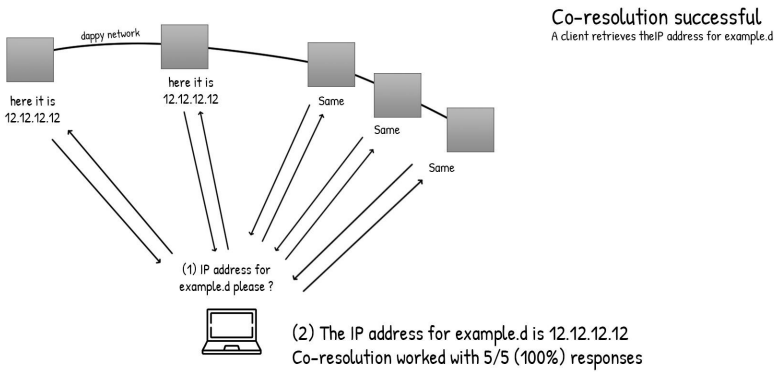
Example 1:

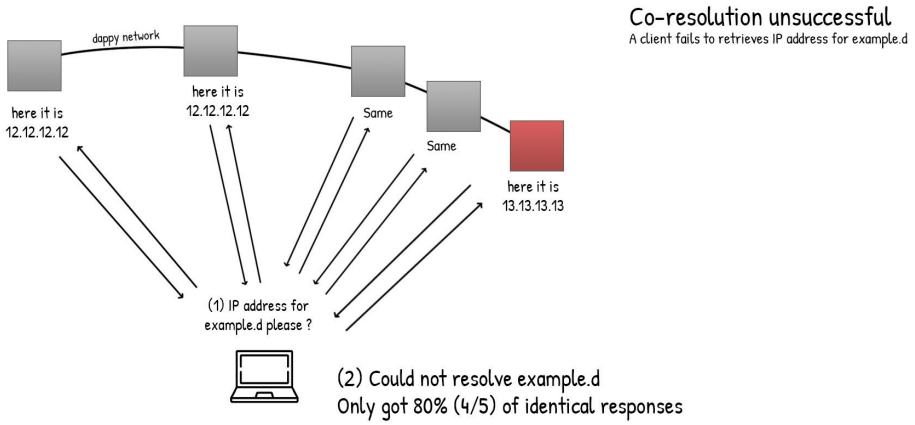
A client knows a dappy network with 10 members, the synchrony constraints are 7 / 100% . When trying to get the IP address or TLS certificate for *example.com*, he will ask that same question *"A records in example.com"* to members until he receives 7 answers (some nodes may be down or not reachable because of network issues). When 7 answers are received, it needs 100% homogeneous answers to go forward.

Example 2:

A client knows a dappy network with 15 members, the criterias are lighter : 5 / 80% . Client will reach many nodes until he gets 5 answers, and then will only require at least 4 out of the 5 responses to agree, before going forward.

Illustrations:





The leaderless state replication system coupled with co-resolution allows web service providers to not rely anymore on any of the many centralized systems they had to work with in the old world, and thus to avoid whole families of attack, reaching new heights in terms of security and privacy with its users.

Attack scenarios that disappear in dappy :

- DNS cache poisoning
- Failed domain verification by a CA
- Mismanagement or hack of a CA
- Mismanagement or hack of a DNS Registry
- Mismanagement or hack og DNS Registrar
- Mismanagement or hack of a DNS resolver service

A web service can be resolved or discovered only if a set of independent companies agree on the addressing and cryptographic data associated with this web service. Dappy network members have trust anchors that identify them, allow clients to reach them over TLS, and through co-resolution they distribute trust anchors for web services, they do not sign the certificates of web services (that are self signed).

Note: Clients will often need at least a TLS certificate and IP address to initiate TLS connection with a web service. One co-resolution may carry many questions, just like DNS messages that can carry two or more questions.

Note: Co-resolution is not a one-by-one rotation system that adds resilience in case one node goes down. Co-resolution does add resilience, but its raison d'être is to distribute trust on independent agents; multiple endpoints operated by different companies must absolutely be used.

Note: This paper wants to stay general and does not provide details about the protocol for co-resolution. DoH is an obvious fit, the only requirements are to use TLS sessions, and not rely on any name system but instead reach the nodes of the dappy network directly with IP address.

Note: Prior to doing co-resolution, clients must have a hardcoded, complete or partial list of the members of the dappy network. Nevertheless, since trust is distributed (no one is trusted), a company of the dappy network is not a certificate authority. Certificate Authorities that don't exist, only domain owners exist, they are sovereign over their TLS identities.

What about DNSSEC and recursion ?

DNS allows recursion (recursive lookups), one authoritative name server may be responsible for *.com*, and point to another DNS authoritative name server (through a NS record) for *example.com*, which can also point to an authoritative server for another subdomain. A company is therefore able to scatter the responsibilities onto different infrastructures or locations on many levels : *japan.example.com, germany.example.com, department1.japan.example.com etc.*

DNSSEC is used precisely in this context, a parent (ex: *example.com*) publishes a trust anchor (public key) for subdomain *example.com*. Later a resolver that needs to resolve A records for *example.com* is able to retrieve the public key from the parent, and verify that the records of *example.com* are properly signed, and not counterfeit data from the authoritative name server, or cache poisoning.

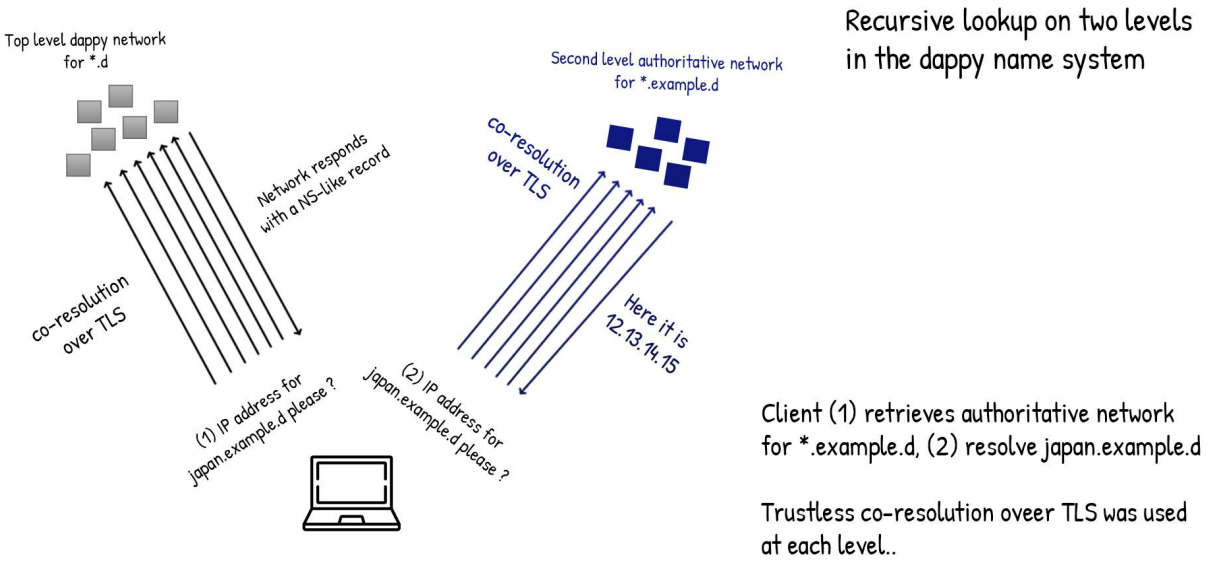
It is possible to do recursion in dappy in a similar fashion to the DNS. A key difference is that dappy relies on co-resolution, and therefore the resolver needs the definition of a network, and not a single IP address or even (IP address + TLS certificate) tuple. it can be implemented in different ways, but essentially, *example.d* which is resolved by the top level dappy network, can point to a second level, privately owned dappy network for *japan.example.d.*

This second level dappy network may be a single node network, it may be a consortium network with many nodes controlled by independent companies just like the top level dappy network.

In DNS a successful lookup never gives trustworthy answers. With the DNS protocol and architecture, DNSSEC is needed because (1), authoritative, a privately owned name server is entirely responsible for a node in the tree, (2) the core DNS protocol is insecure and without encryption.

A successful co-resolution in dappy is trustworthy and returns accurate data, except under extraordinary unlikely circumstances (network wide hack). **With dappy protocol and architecture, (1) authoritative name servers are replaced by authoritative co-resolution networks, (2) every co-resolution is over TLS. DNSSEC, any additional PKI or recursive signature of records is not needed.**

DNSSEC is very complex to set up, it includes many new record types (DNSKEY, DS, NSEC3, RRSIG), an additional PKI and typically stops at the DNS resolver level, DoH must therefore be used in addition by clients. The way dappy establishes and maintains trust over the hierarchy does not use new record types, or additional signature systems, it is simpler, lighter and goes down to the client or browser level.



Scoping under a new TLD

We argue that trying to articulate existing DNS domain names (*.com*, *.net* etc.) with dappy is a bad idea, as the dappy protocol is too different. It could lead to heterogeneous implementations and inaccuracies. How should the browser or another program know if *example.com* is to be looked up in the DNS + CA system, or in the dappy name system ? How should we map the ownerships in the private databases of the DNS, with the state replication system that only accepts cryptography as a proof of ownership (see paragraph on censorship for more precise take on this).

Scoping under a new TLD resolves those issues, it prevents any collision with existing DNS domains, allows programs and developers to explicitly tell browsers or other programs to do co-resolution, and also gives users a visual indication of the high security standards that were applied for setting up the connection.

Note: dappy mainnet has chosen *.d* as the TLD under which the domains are scoped.

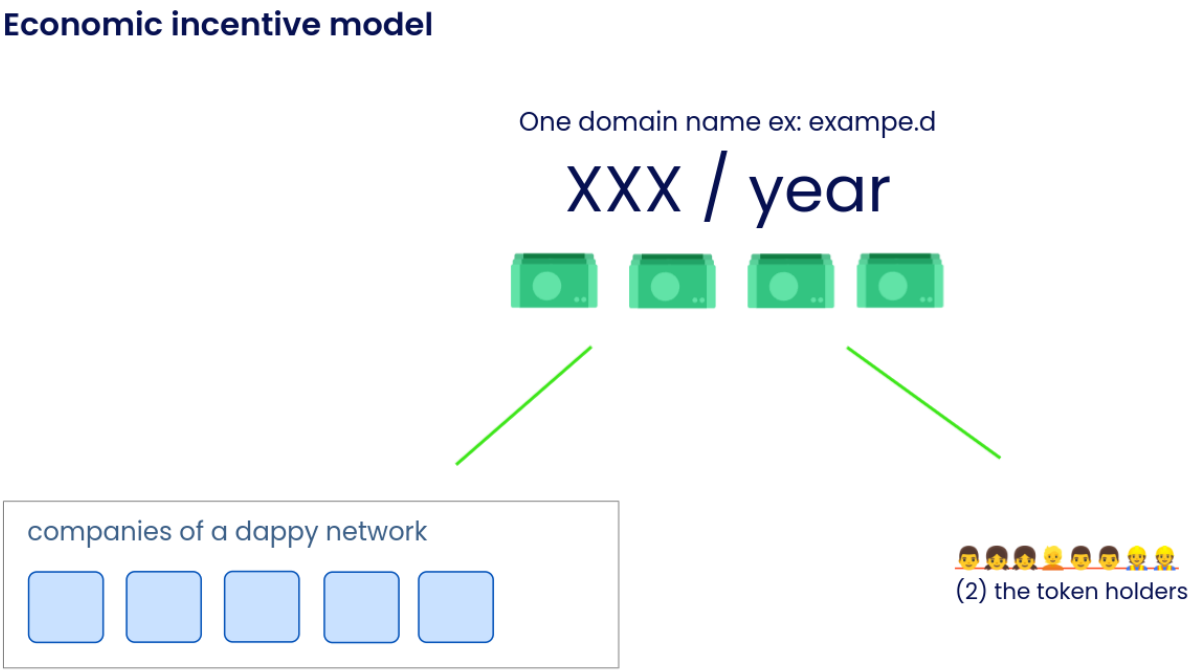
Economic incentive system

Although it is not a core technical feature, dappy must embed an economic incentive system. Companies and organizations need financial compensation for participating in the security of the network, responding to requests and storing the zones of the name system.

Domain names need to be purchased, and renewed every year, and money coming from the purchases can be equally shared among the members of the dappy network. The base price can be decided collectively to match strategic goals regarding adoption and/or return on investment. Users would then be free to sell domains using marketplaces that may be developed by one, or many members of the dappy network, or even independently of the dappy network.

Note: The current, main implementation of dappy chose the no-token no-gas approach. With a PoS blockchain, the usage of a token would have been

obvious. In the present case, security of the protocol is not tokenized, and users don't need any token to update their DNS zone. However, a token can be useful, and allow a shared governance model as it adds new actors for governance of the protocol (in addition to network members), it can allow other actors to earn rewards (see illustration), and eventually unlock special features or discounts. The repartition of dappy network vs token for reward or governance, or both may be 80%/20%, 50%/50% etc.



Note: Today very few companies provide free DNS resolution service, they know everything about the web services that are accessed by certain IP addresses that they could map to users, companies or countries. The cause of this is that DNS resolution is not at all connected to the economics of the DNS, companies have no direct interest or incentive to provide DNS resolution services for free. Dappy solves this problem, the companies that process the payments and store the domains are the same that resolve and provide public infrastructure for service discovery, and there are many more than two or three. In addition, co-resolution may include randomization, so that different endpoints are reached for each co-resolution.

4 - Details and implementation discussions

Non tokenized security

Dappy needs a leaderless state replication mechanism to work, in which members are independent and only trust messages (or transactions) that are cryptographically signed for state changes. This state replication system could in theory be a proof of work blockchain, or proof of stake, it could be permissioned or permissionless as well.

Permissionless proof of stake blockchains chose to tokenize the security of the protocols, it may be a fit for general purpose smart contract platform. We argue that for dappy, a permissioned network without tokens makes more sense, by “without taken” (or no-token), we mean that the security or power over the state is not weighted by tokens, but equally split, each member has a weight of one.

This second option makes sense because in web browsers, you have to ship a finite list of trust anchors, it cannot be ever evolving like the stakes repartition in a PoS blockchain. The no-token approach also allows free updates during the validity period of a domain, users only pay once a year, then they are free to update their zone, add subdomains and TLS certificates with no token or gas fee, they just need the private key to sign the messages. Although there may be some concern about DDOS protection, being harder to achieve (or at least not inherent to the protocol) the no-token approach, it is a great improvement in user experience, and makes everything simpler for domain management.

Note: It is important to understand that a name system is very different from a general purpose smart contract platform, because some resources are guaranteed to never interact with one another : transactions that change the TLS certificate (CERT record) of *example.d* don’t need to be strictly ordered vis-à-vis transactions that add an A record for *example2.d* . Just like a sharded blockchain or database, computations for independent domains can be massively parallelized with no risk.

Note: At the time this paper is written, dappy team chose the second option : permissioned network and yearly-based renewals with no gas fee or token.

Visibility or monitoring benefits

Putting trust anchors back in the name system (DANE proposition) brings hidden and non-obvious benefits.

Now administrators and security engineers can have instant visibility over all the TLS certificates linked to or under a given domain or subdomain, internet-wide. This is possible because TLS certificates belong to the name system, which is a tree.

The horizontal structure of the web PKI prevents that, as one would have to guess which web PKI their clients use, and query each certificate authority (no standardized protocol exists for that).

Fiat gateway, or smart contract

A leaderless state replication mechanism works great - once a user has already entered - what could be the gateway to dappy ? Should it allow fiat payment ?

A solution (solution 1) could be to rely on a stablecoin in a major smart contract platform, a user would have to send cryptocurrency to a specific smart contract, specifying the domain he/she wants in the transaction metadata. This way revenues could be automatically shared, thanks to the trustless and transparent nature of blockchain computations.

A fiat gateway where credit card payment is used can be easier to use for common people or companies (solution 2). With this configuration, members of the dappy network would trust one to issue the domain names once they are paid for. Some may argue that this is flawed because it becomes a leaderful system, it is only half true. A domain that has already been purchased can not be corrupted by the leader that manages the gateway (the leader does not have private key linked to domains), the pseudo-leader only has a unique prerogative for new domains issuance.

Note: as it is still in 2022 very early and simplifies adoption, dappy team chose to go with option 2 and a fiat payment gateway operated by one company, this is again not inherent to the system, transition to crypto can happen seamlessly.

Client to node, which protocol to use ?

DNS clients today (browsers or operating systems) know a DNS resolver's IP address, this IP address is the trust anchor for the DNS, it can be hardcoded or configurable. In a dappy context a client, program or web browser has the definition of a dappy network. A definition includes IP address(es), hostnames and self signed TLS certificates for each member, network definition is a sort of trust anchor.

A dappy zone could be the exact same as a DNS zone, with just new records for TLS certificates. In this context DNS-over-HTTPS (DoH) can be the best way to go for client to node communication, it articulates well with the record-based structure of the DNS zones. Other protocols or custom routes may be used for service discovery, of course the only requirement is that they use a secure TLS encryption.

A new record type

DANE introduced a new record type in the DNS zones : TLSA. Dappy introduces a new one as well, the CERT record type. Obviously, a CERT record contains the root certificate for a given domain, or subdomain. This is however an implementation detail, it may eventually change.

Compatibility with web browsers

Major web browsers use a single endpoint for DNS (or DoH) queries, cannot do, and don't know about co-resolution. The web PKI side also is a one authority system. We described those two protocols in Limitation and security flaw number one : web PKI and Limitation and security flaw number three : DNS.

Major browsers don't speak dappy, they are not compatible. Scoping under a new TLD is here an important feature, browser vendors that will implement dappy can naturally switch the IP address lookup and certificate verification process, based on the TLD, and therefore not break anything, or trigger domain name collisions.

Censorship or unilateral decision

Note: Many applications powered by blockchain, or blockchain name systems claim to be censorship resistant. Let's assume PoW or PoS are security mechanisms that are censorship resistant, then layer 1 is censorship resistant. But from a light client perspective (almost every computer or smartphone is a light client) they are not at all (see Point of view matters). Light clients often use a single entry point. It just takes one blockchain node to send fake data for the clients to be fooled. As an example, some big web3 infrastructure companies provide an abstraction layer, and fast read/write access to the blockchain for thousands of dapps and web3 services. For a public name system, or public web PKI, this cannot work at all.

Instead of censorship resistance, we prefer to phrase it the following way :

Co-resolution over a leaderless state replication system removes the impact of unilateral decisions. Censorship, domain removal can still happen, but only as the result of collective decisions. Depending on the client settings, it may be a decision of 20%, 30% or 50% of the dappy network that breaks co-resolution and leads to a domain being unusable through dappy.

Brand protection, law enforcement

As it is explained in the previous paragraph, what dappy changes fundamentally, is the impact of unilateral decisions, it disappears. Censorship, or any operation that can be assimilated to brand protection like domain blocking or confiscation can not occur without agreement of the majority of the dappy network, which ideally spreads across various jurisdictions.

Every online business, SaaS or web application that conducts legal, non-subversive service will greatly benefit from this decentralized network securing the service discovery.

Should dappy have a brand protection program ? Should network members remove cybersquatting domains, should they confiscate a domain if it has been stolen from its legitimate owner ? How should we determine the legitimate owner ?

The core dappy protocol does not take in consideration any of those questions, it is up to the members to take decisions collectively or single-handedly based on their legal obligation, jurisdiction or moral motives.

What we can say is that it could be strategic to incorporate a level of brand protection, if *Example* is a multi-Billion dollars company, *example.d* probably deserves to be given back to the legitimate owners in case of theft. The dappy network may agree on setting up such practices. Cybersquatting is another issue that is more complex to tackle as it often includes thousands of domains.

When dappy launched (september 2022), it launched with a reserved list of 20.000+ domain names (second level), 1500 ICANN TLDs and 500 international brands, that can of course be claimed.

Junctions, and CSP at name system level

Other explorations and novelties above the core dappy protocol will be discussed in other papers. We still want to give you a glimpse of what can be added or built upon the core dappy protocol.

Junctions (or Junctions domains) is a new syntax in the search bar of the browser (ex: *“bob.d & alice.d”*), that allows web services to be endorsed by many identities instead of one, without any trusted intermediary.

Storing *Content-Security-Policy* headers in the name system as TXT records could be game changing for many organizations, it is another draft proposal that was born inside the dappy project. This feature allows the name system

to define the rules that secure the execution of web applications (where images can be loaded from, what are the authorized external links a user may navigate to etc.) and take precedence over the web server (back end). A company responsible for 100 domains can now administrate the CSP rules at the name system level, delegate the development to other services or external companies while keeping control over the security rules for client-side execution.

5 - Conclusion

Many implementation details have been voluntarily ignored in the present paper as we want to focus on the essential components of dappy.

The massive deployment of certificate verification services, TLS and the integration of those technologies in web browsers allowed for a whole new category of web services to emerge. This paper proposes a new way for securing identities and doing service discovery for public web services, with superior levels of security, privacy and censorship resistance.

We hope the reader sees that the assemblage of technologies dappy proposes - centered on distributed trust - has the potential to avoid dozens of cyberattacks scenarios by securing today's critical web services, and in addition to allow a new generation of privacy-focused, decentralized, and disruptive public web services to emerge as well.

(1) Eric Rescola, December 28th 2021, https://educatedguesswork.org/posts/dns-security-dane/
(2) P. Hoffman, VPN Consortium, J. Schlyter, Kirei AB, RFC 6698 https://www.rfc-editor.org/rfc/rfc6698 August 2012
(3) Trust me, I'm a Root CA! Analyzing SSL Root CAs in modern Browsers and Operating Systems. Tariq Fadai, Sebastian Schrittwieser, Peter Kieseberg, Martin Mulazzani https://publications.sba-research.org/publications/SSL.pdf
(4) Eric Rescola, November 25th 2022, https://educatedguesswork.org/posts/eidas-article45/

(5) Palisade Consulting, November 2021, https://palisade.consulting/blog/tld-hacking
(6) RFC 6962 https://www.rfc-editor.org/rfc/rfc6962.txt
(7) Wired, April 2017 https://www.wired.com/2017/04/hackers-hijacked-banks-entire-online-operation/
(7) C. Ellison and B. Schneier. Ten Risks of PKI: What You're not Beeing Told about Public Key Infrastructure. Computer Security Journal, 16, 2000 https://www.schneier.com/academic/archives/2000/01/ten_risks_of_pki_wha.html