A Censorship-Resistant, Privacy-Enhancing and Fully Decentralized Name System

Matthias Wachs, Martin Schanzenbach, Christian Grothoff

Network Architectures and Services
Fakultät für Informatik
Technische Universität München
{wachs,schanzen,grothoff}@in.tum.de

Abstract. The Domain Name System (DNS) is vital for access to information on the Internet. This makes it a target for attackers whose aim is to suppress free access to information. This paper introduces the design and implementation of the GNU Name System (GNS), a fully decentralized and censorship-resistant name system. GNS provides a privacy-enhancing alternative to DNS which preserves the desirable property of memorable names. Due to its design, it can also double as a partial replacement of public key infrastructures, such as X.509. The design of GNS incorporates the capability to integrate and coexist with DNS. GNS is based on the principle of a petname system and builds on ideas from the Simple Distributed Security Infrastructure (SDSI), addressing a central issue with the decentralized mapping of secure identifiers to memorable names: namely the impossibility of providing a global, secure and memorable mapping without a trusted authority. GNS uses the transitivity in the SDSI design to replace the trusted root with secure delegation of authority, thus making petnames useful to other users while operating under a very strong adversary model. In addition to describing the GNS design, we also discuss some of the mechanisms that are needed to smoothly integrate GNS with existing processes and procedures in Web browsers. Specifically, we show how GNS is able to transparently support many assumptions that the existing HTTP(S) infrastructure makes about globally unique names.

1 Introduction

The Domain Name System (DNS) is a unique distributed database and a vital service for most Internet applications. While DNS is distributed, it relies on centralized, trusted registrars to provide globally unique names. As the awareness of the central role DNS plays on the Internet rises, various institutions are using their power (including legal means) to engage in attacks on the DNS, thus threatening the global availability and integrity of information on the Web [1]. This danger has also been recognized by the European Parliament, which has emphasized the importance of maintaining free access to information on the Web in a resolution [2]. Tampering with the DNS can cause collateral damage, too: a recent study [3] showed that Chinese censorship of the DNS has had worldwide effects on name resolution. At the same time, we observe that the Internet's importance for free communication has dramatically risen: the events of the Green Revolution in Iran and the Arab Spring have demonstrated this. Dissidents need communication channels that provide the easy linking to information that is at the Web's core. This calls for a censorship-resistant name system which ensures that names of Internet servers can always be resolved correctly.

DNS was not designed with security as a goal. This makes it very vulnerable, especially to attackers that have the technical capabilities of an entire nation state at their disposal. The follow are some of the most severe weaknesses that the DNS exhibits even in the presence of the DNS Security Extensions (DNSSEC). DNSSEC [4] was designed to provide data integrity and origin authentication to DNS. DNSSEC maintains the hierarchical structure of DNS and thus places extensive trust in the root zone and TLD operators. More importantly, DNSSEC fails to provide any level of query privacy [5]: the content of DNS queries and replies can be read by any adversary with access to the communication channel and can subsequently be correlated with users. On a technical level, current DNSSEC deployment suffers from the use of the RSA crypto system, which leads to large key sizes. This can result in message sizes that exceed size restrictions on DNS packets, leading to additional vulnerabilities [6]. Finally, DNSSEC is not designed to withstand legal attacks. Depending on their reach, governments, corporations and their lobbies can legally compel operators of DNS authorities to manipulate entries and certify the changes, and Soghoian and Stamm have warned that similar actions might happen for X.509 server certificates [7]. There can also be collateral damage: DNSSEC cannot prevent problems such as the recent brief disappearance of thousands of legitimate domains during the execution of established censorship procedures, in which the Danish police accidentally requested the removal of 8,000 (legitimate) domain names from DNS and providers complied. The underlying attack vector in these cases is the same: names in the DNS have owners, and ownership can be taken away by different means.

This paper presents the GNU Name System (GNS), a censorship-resistant, privacy-preserving and decentralized name system designed to provide a secure alternative to DNS, especially when censorship or manipulation is encountered. As GNS can bind names to any kind of cryptographically secured token, it can double in some respects as an alternative to some of today's Public Key Infrastructures, in particular X.509 for the Web.

The foundation of the GNS system is a petname system [8], where each individual user may freely and securely map names to values. In a petname system, each user chooses a *nickname* as his preferred (but not necessarily globally unique) name. Upon introduction, users adopt the nickname by default as a *label* to refer to a new acquaintance; however, they are free to select and assign any *petname* of their choice in place of—or, in addition to—the nickname. Petnames thus reflect the personal choice of the individual using a name, while nicknames are the preferred name of the user that is being identified.

The second central idea is to provide users with the ability to securely delegate control over a subdomain to other users. This simple yet powerful mechanism is borrowed from the design of SDSI/SPKI. With the combination of petname system and delegation, GNS does not require nor depend on a centralized or trusted authority, making the system robust against censorship attempts. Decentralization and additional censorship resistance is achieved by using a distributed hash table (DHT) to enable the distribution and resolution of key-value mappings. In theory, any DHT can be used. However, depending on the properties of the DHT in question, varying degrees of censorship resistance will be the result. As such, the choice of the DHT is crucial to the system. Finally, GNS is privacy-preserving since both key-value mappings as well as queries and responses are encrypted such that an active and participating adversary can at best perform a confirmation attack, and can otherwise only learn the expiration time of a response.

While this combination yields a secure name system, it also violates a fundamental assumption prevailing on the Web, namely that names are globally unique. Thus, together with the working implementation of GNS[1], another key contribution of our work is the construction of system components to enable the use of GNS in the context of the Web. We provide ready-to-use components to enable existing Web applications to use GNS (and DNS in parallel, if desired) without any prior modifications and knowledge.

As a alternative public key infrastructure, GNS can also be combined with existing PKI approaches (such as X.509, DANE, Tor's ".onion" or the Web of Trust) to either provide memorable names or alternative means for verification with increased trust agility. In combination with TLSA records, GNS can replace existing X.509 certification authorities as described in Appendix A.3.

2 Background

In order to present GNS, we must first discuss technical background necessary to understand our design. We define the adversary model that GNS addresses and then provide some brief background on DNS, DNSSEC, SDSI/SPKI and distributed storage in P2P Networks.

2.1 Adversary Model

The adversary model used in this work is modeled after a state trying to limit access to information without causing excessive damage to its own economy. The goal of the adversary is to force name resolution to change in the respective name system, by either making the resolution fail or by changing the value to which an existing name (not originally under the control of the adversary) maps.

We allow the adversary to participate in any role in the name system. Note that this excludes the possibility of a global trusted third party. In addition, the adversary is allowed to assume multiple identities. We impose no bound on the fraction of collaborating malicious participants, and we assume that the

[1] Available under: `https://gnunet.org/gns`

adversary can take control of names using judicial and executive powers (for example by confiscating names or forcing third parties to misdirect users to adversary-controlled impostor sites). Computationally, the adversary is allowed to have more resources than all benign users combined.

The adversary may directly compromise the computers of individual users; for the security of the system as a whole, we only require that the impact of such attacks remains localized. The rationale for being able to make such an assumption is that the economic and political cost of such tailored methods is very high, even for a state actor. Similarly, the adversary cannot prevent the use of cryptography, free software, or encrypted network communication. The adversary is assumed to be unable to break cryptographic primitives. As far as network communication is concerned, we assume that communication between benign participants generally passes unhindered by the adversary.

Zooko's triangle [9], an insightful conjecture that is often used to define the possible design space of name systems, has important implications under this adversary model: it means that no name system can provide globally unique and memorable names and be secure [10]. It should be noted that in weaker adversary models, these implications do not hold [11].

2.2 DNS and DNSSEC

The Domain Name System is an essential part of the Internet as it provides mappings from host names to IP addresses, providing memorable names for users. DNS is hierarchical and stores name-value mappings in so-called *records* in a distributed database. A record consists of a name, type, value and expiration time. Names consist of *labels* delimited by dots. The root of the hierarchy is the empty label, and the right-most label in a name is known as the top-level domain (TLD). Names with a common suffix are said to be in the same *domain.* The *record type* specifies what kind of value is associated with a name, and a name can have many records with various types. The most common record types are "A" records that map names to IPv4 addresses.

The Domain Name System database is partitioned into *zones.* A *zone* is a portion of the namespace where the administrative responsibility belongs to one particular authority. A zone has unrestricted autonomy to manage the records in one or more domains. Very importantly, an authority can delegate responsibility for particular *subdomains* to other authorities. This is achieved with an "NS" record, whose value is the name of a DNS server of the authority for the subdomain. The *root zone* is the zone corresponding to the empty label. It is managed by the Internet Assigned Numbers Authority (IANA), which is currently operated by the Internet Corporation for Assigned Names and Numbers (ICANN). The National Telecommunications and Information Administration (NTIA), an agency of the United States Department of Commerce, assumes the (legal) authority over the root zone. The root zone contains "NS" records which specify names for the authoritative DNS servers for all TLDs.

The Domain Name System Security Extensions add integrity protection and data origin authentication for DNS records. DNSSEC does not add confidentiality nor denial-of-service protection. It adds record types for public keys ("DNSKEY") and for signatures on resource records ("RRSIG"). DNSSEC relies on a hierarchical public-key infrastructure in which all DNSSEC operators must participate. It establishes a trust chain from a zone's authoritative server to the trust anchor, which is associated with the root zone. This association is achieved by distributing the root zone's public key out-of-band with, for example, operating systems. The trust chains established by DNSSEC mirror the zone delegations of DNS. With TLD operators typically subjected to the same jurisdiction as the domain operators in their zone, these trust chains are at risk of attacks using legal means.

2.3 SDSI/SPKI

SDSI/SPKI is a merger of the Simple Distributed Security Infrastructure (SDSI) and the Simple Public Key Infrastructure (SPKI) [12]. It defines a public-key infrastructure that abandons the concept of memorable global names and does not require certification authorities. SDSI/SPKI has the central notion of *principals*, which are globally unique public keys. These serve as namespaces within which local names are defined. A name in SDSI/SPKI is a public key and a local identifier, e.g. $K - Alice$. This name defines the identifier *Alice*, which is only valid in the namespace of key K. Thus, $K_1 - Alice$ and $K_2 - Alice$ are different names. SDSI/SPKI allows namespaces to be linked, which results in compound names: $K_{Carol} - Bob - Alice$ is Carol's name for the entity which Bob refers to as $K_{Bob} - Alice$. Bob himself is identified by Carol as $K_{Carol} - Bob$. SDSI/SPKI allows assertions about names by issuing certificates[2]. A *name cert* is a tuple of *(issuer public key, identifier, subject, validity)*, together with a signature by the issuer's private key. The *subject* is usually the key to which a name maps. Compound names are expressed as certificate chains.

GNS applies these key ideas from SDSI/SPKI to a name resolution mechanism in order to provide an alternative to DNS. The transitivity at the core of SDSI/SPKI is found in GNS as *delegation of authority* over a name. In both systems, name resolution starts with a lookup in the local namespace.

2.4 Distributed Storage in P2P Overlay Networks

In peer-to-peer systems, it is common to use a DHT to exchange data with other participants in the overlay. A DHT creates a decentralized key/value store to make mappings available to other users and to resolve mappings not available locally. GNS uses a DHT to make local namespace and delegation information available to other users and to resolve mappings from other users. As mentioned previously, the choice of DHT strongly affects the availability of GNS data.

[2] Ultimately, SDSI/SPKI allows to create authorizations based on certificates and is a flexible infrastructure in general, but we will focus only on the names here.

3 Design of the GNU Name System

In the following, we describe the core concepts of GNS that are relevant to users. The cryptographic protocol used to ensure query privacy is explained in Section 4, and the protocol for key revocation in Section A.5.

3.1 Names, Zones and Delegations

GNS employs the same notion of names as SDSI/SPKI: principals are public keys, and names are only valid in the local namespace defined by that key. Namespaces constitute the *zones* in GNS: a zone is a public-private key pair and a set of records. GNS records consist of a label, type, value and expiration time. Labels have the same syntax as in DNS; they are equivalent to local identifiers in SDSI/SPKI. Names in GNS consists of a sequence of labels, which identifies a *delegation path.* Cryptography in GNS is based on elliptic curve cryptography and uses the ECDSA signature scheme with Curve25519 [13].

We realise a petname system by having each user manage his own zones, including, in particular, his own personal *master zone.*[3] Users can freely manage mappings for memorable names in their zones. Most importantly, they can delegate control over a subdomain to another user (which is locally known under the petname assigned to him). To this end, a special record type is used (see Section 3.5). This establishes the aforementioned delegation path. Each user uses his master zone as the starting point for lookups in lieu of the root zone from DNS. For interoperability with DNS, domain names in GNS use the pseudo-TLD ".gnu". ".gnu" refers to the GNS master zones (i. e. the starting point of the resolution). Note that names in the ".gnu" pseudo-TLD are always relative.

Publishing delegations in the DHT allows transitive resolution by simply following the delegation chains. Records can be private or public, and public records are made available to other users via a DHT. Record validity is established using signatures and controlled using expiration values. The records of a zone are stored in a *namestore* database on a machine under the control of the zone owner.

We illustrate the abstract description above with the example shown in Figure 1. The figure shows the paths Alice's GNS resolver would follow to resolve the names "www.dave.carol.gnu" and "www.buddy.bob.gnu", both of which refer to Dave's server at IP "192.0.2.1". For Carol, Dave's server would simply be "www.dave.gnu". It is known to Alice only because both Bob and Carol have published public records indicating Dave, and Alice can resolve the respective delegation chain via her known contacts. Recall that zones are identified using public keys and records must be cryptographically signed to ensure authenticity and integrity.

[3] Each user can create any number of zones, but must designate one as the master zone.

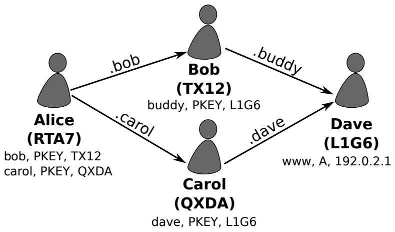


Fig. 1. Name resolution graph in GNS. Each user is shown with a fingerprint of his master zone and the public records from this zone in the format *name, type, value.*

3.2 Zone Management with Nicknames and Petnames

Suppose Alice runs a web server and wants to make it available with GNS. In the beginning she sets up her master zone using GNS. After the public-private key pair is generated, Alice can create a revocation notice to be able to immediately revoke their GNS zone in case she gets compromised. Suppose Alice wants to propose that her preferred nickname is “carol” to other users. She therefore uses the new “NICK” record that GNS provides. For her web server, she creates an appropriate public “A” record under the name “www”. This “A” record is the same as in DNS. To make it resolvable by other users, this record is marked as public and published in the DHT.

Now suppose we have a second user, Bob. He performs the same setup on his system, except that his preferred nickname is just “bob”. Bob gets to know Alice in real life and obtains her public key. To be able to contact Alice and access her web server, he then adds Alice to his zone by adding a new delegation using the new “PKEY” record. Bob can choose any name for Alice’s zone in *his* zone. Nevertheless, Bob’s software will default to Alice’s preferences and suggest “carol”, as long as “carol” has not already been assigned by Bob. This is important as it gives Alice an *incentive* to pick a nickname that is (sufficiently) unique to be available among the users that would delegate to her zone. By adding Alice’s public key under “carol”, Bob delegates queries to the “*.carol.gnu” subdomain to Alice. Thus, from Bob’s point of view, Alice’s web server is “www.carol.gnu”. Note that there is no need for Alice’s nickname “carol” to be globally unique, they should only not already be in use within Alice’s social group.

3.3 Relative Names for Transitivity of Delegations

Users can delegate control over a subdomain to another user’s zone by indicating this in a new record, “PKEY”. Suppose Dave is Bob’s friend. Dave has added a delegation to Bob with a “PKEY” record under the name “buddy”—ignoring Bob’s preference to be called “bob”. Now suppose Bob wants to put on his webpage a link to Alice’s webpage. For Bob, Alice’s website is “www.carol.gnu”. For Dave, Bob website is “buddy.gnu”. Due to delegation, Dave can access Alice’s website under “www.carol.buddy.gnu”. However, Bob’s website cannot contain that link: Bob may not even know that he is “buddy” for Dave.

We solve this issue by having Bob use “www.carol.+” when linking to Alice’s website. Here, the “+” stands for the originating zone. When Dave’s client encounters “+” at the end of a domain name, it should replace “+” with the name of the GNS authority of the site of origin. This mechanism is equivalent to relative URLs, except that it works with hostnames.

3.4 Absolute Names

In GNS, the “.gnu” pseudo-TLD is used to provide secure and memorable names which are only defined relative to some master zone. However, introducing new zones into the system ultimately requires the ability to reference a zone by an absolute identifier, which must correspond to the public key of the zone. To facilitate dealing with public keys directly, GNS uses the pseudo-TLD “.zkey”, which indicates that the specified domain name contains the public key of a GNS zone. As a result, the “.zkey” pseudo-TLD allows users to use secure and globally unique identifiers. Applications can use the “.zkey” pseudo-TLD to generate a domain name for a GNS zone for which the user does not (yet) have a memorable name. A label in the “.zkey” pseudo-TLD is the public key of the zone encoded within the 63 character limitations for labels imposed by DNS.

3.5 Records in GNS

As GNS is intended to coexist with DNS, most DNS resource records from [14,15] (e. .g., “A”, “MX”) are used with identical semantics and binary format in GNS. GNS defines various additional records to support GNS-specific operations. These records have record type numbers larger than 2^{16} to avoid conflicts with DNS record types that might be introduced in the future. Details on all record types supported by our current implementation can be found in our technical report [16].

4 Query Privacy

To enable other users to look up records of a zone, all public records for a given label are stored in a cryptographically signed block in the DHT. To maximize user privacy when using the DHT to look up records, both queries and replies are encrypted. Let $x \in \mathbb{Z}_n$ be the ECDSA private key for a given zone and $P = xG$ the respective public key where G is the generator of the elliptic curve. Let $n := |G|$ and $l \in \mathbb{Z}_n$ be a numeric representation of the label of a set of records $R_{l,P}$. Using

$$h : = x \cdot l \mod n \tag{1}$$

$$Q_{l,P} : = H(hG) \tag{2}$$

$$B_{l,P} : = S_h(E_{\mathtt{HKDF}(l,P)} R_{l,P}), hG \tag{3}$$

we can then publish $B_{l,P}$ under $Q_{l,P}$ in the DHT, where S_h represents signing with the private key h, HKDF is a hash-based key derivation function and E

represents symmetric encryption based on the derived key. Any peer can validate the signature (using the public key hG) but not decrypt $B_{l,P}$ without knowledge of both l and P. Peers knowing l and P can calculate the query

$$Q_{l,P} = H(lP) = H(lxG) = H(hG) \tag{4}$$

to retrieve $B_{l,P}$ and then decrypt $R_{l,P}$.

Given this scheme, an adversary can only perform a confirmation attack; if the adversary knows both the public key of the zone and the specific label, he can perform the same calculations as a peer performing a lookup and, in this specific case, gain full knowledge about the query and the response. As the DHT records are public, this attack cannot be prevented. However, users can use passwords for labels to restrict access to zone information to authorized parties. The presented scheme ensures that an adversary that is unable to guess both the zone's public key and the label cannot determine the label, zone or record data.

5 Security of GNS

One interesting metric for assessing the security of a system is to look at the size of the trusted computing base (TCB). In GNS, users explicitly see the trust chain and thus know if the resolution of a name requires trusting a friend, or also a friend-of-a-friend, or even friends-of-friends-of-friends—and can thus decide how much to trust the result. Naturally, the TCB for all names can theoretically become arbitrarily large—however, given the name length restrictions, for an individual name it is always less than about 125 entities. The DHT does not have to be trusted; the worst an adversary can do here is reduce performance and availability, but not impact integrity or authenticity of the data.

For DNS, the size of the TCB is first of all less obvious. The user may think that only the operators of the resolvers visible in the name and their local DNS provider need to be trusted. However, this is far from correct. Names can be expanded and redirected to other domains using "CNAME" and "DNAME" records, and resolving the address of the authority from "NS" records may require resolving again other names. Such "out-of-bailiwick" "NS" records were identified as one main reason for the collateral damage of DNS censorship by China [3]. requires correct information from "x.gtld-servers.net" (the authority for ".com"), which requires trusting "X2.gtld-servers.net" (the authority for ".net"). While the results to these queries are typically cached, the respective servers must be included in the TCB, as incorrect answers for any of these queries can change the ultimate result. Thus, in extreme cases, even seemingly simple DNS lookups may depend on correct answers from over a hundred DNS zones [17]; thus, with respect to the TCB, the main difference is that DNS is very good at obscuring the TCB from its users.

In the following, we discuss possible attacks on GNS within our adversary model. The first thing to note is that as long as the attacker cannot gain direct control over a user's computer, the integrity of master zones is preserved. Attacks on GNS can thus be classified in two categories: attacks on the network, and attacks on the delegation mechanism.

Attacks on the network can be staged as Eclipse attacks. The success depends directly on the DHT. Our choice, R^5N, shows a particularly good resistance against such attacks [18].

Concerning the delegation mechanism, the attacker has the option of tricking a user into accepting rogue mappings from his own zones. This requires social engineering. We assume that users of an anti-censorship system will be motivated to carefully check whose mappings they trust. Nevertheless, if the attacker succeeds, some damage will be done: all users that use this mapping will be affected. The effect thus depends on the "centrality" of the tricked user in the GNS graph. It is difficult to give estimates here, as the system is not deployed yet. In order to maximize the effects of his attack, the attacker would have to carry out his social engineering many times, which is naturally harder. Comparing this to DNSSEC, we note that even when a compromise has been detected, DNS users cannot choose whose delegations to follow. In GNS, they can attempt to find paths in the GNS graph via other contacts. The system that is most similar and in deployment is the OpenPGP Web of Trust. Ulrich et al. found that the Web of Trust has developed a strong mesh structure with many alternative paths [19]. If GNS develops a similar structure, users would greatly benefit.

Finally, censorship does not stop with the name system, and for a complete solution we thus need to consider censorship at lower layers. For example, an adversary might block the IP address of the server hosting the critical information. GNS is not intended as an answer to this kind of censorship. Instead, we advocate using tools like Tor [20] to circumvent the blockade.

6 Related Work

Timeline-based systems in the style of Bitcoin [21] have been proposed to create a global, secure and memorable name system [11]. Here, the idea is to create a single, globally accessible timeline of name registrations that is append-only. In the Namecoin system [22], a user needs to expend computational power on finding (partial) hash collisions in order to be able to append a new mapping. This is supposed to make it computationally infeasible to produce an alternative valid timeline. It also limits the rate of registrations. However, the Namecoin system is not strong enough in our adversary model, as the attacker has more computational power than all other participants, which allows him to create alternative valid timelines. Note that our adversary model is not a far-fetched assumption in this context: it is conceivable that a nation-state can muster more resources than the small number of other entities that participate in the system, especially for systems used as an alternative in places where censorship is encountered or during the bootstrapping of the network, when only a small number of users participate.

The first practical system that improves confidentiality with respect to DNS queries and responses was DNSCurve [5]. In DNSCurve, session keys are exchanged using Curve25519 [13] and then used to provide authentication and encryption between caches and servers. DNSCurve improves the existing Domain Name System with confidentiality and integrity, but the fundamental issues of DNS with respect to the adversary trying to modify DNS mapping is not within its focus.

GNS has much in common with the name system in the Unmanaged Internet Architecture (UIA) [23], as both systems are inspired by SDSI. In UIA, users can define personal names bound to self-certifying cryptographic identities and can access namespaces of other users. UIA's focus is on universal connectivity between a user's many devices. With respect to naming, UIA takes a clean-slate approach and simply assumes that UIA applications use the UIA client library to contact the UIA name daemon and thus understand the implications of relative names. In contrast, GNS was designed to interoperate with DNS as much as possible, and we have specifically considered what is needed to make it work as much as possible with the existing Internet. In terms of censorship resistance, both systems inherit basic security properties from SDSI with respect to correctness.

7 Summary and Conclusion

GNS is a censorship resistant, privacy-enhancing name system which avoids the use of trusted third parties. GNS provides names that are memorable, secure and transitive. Placing names in the context of each individual user eliminates ownership and effectively eliminates the possibility of executive or judicial control over these names.

GNS can be operated alongside DNS and begins to offer its advantages as soon as two parties using the system interact, enabling users to choose GNS or DNS based on their personal trade-off between censorship-resistance and convenience.

GNS and the related tools are available to the public as part of the GNUnet peer-to-peer framework and are free software under the GNU General Public License. The current implementation includes all of the features described in this paper. In the future, we will begin deployment to actual users and perform experiments to find out which usability problems arise with GNS.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG) under ENP GR 3688/1-1. We thank Kenneth Almquist, Jacob Appelbaum, Daniel Bernstein, Ludovic Courtès, Tanja Lange, Luke Leighton, Simon Josefsson, Nikos Mavrogiannopoulos, Ondrej Mikle, Stefan Monnier, Niels Möller, Chris Palmer, Martin Pool, Richard Stallman, Neal Walfield and Zooko Wilcox-O'Hearn for insightful comments and discussions on an earlier draft of the paper. We thank Florian Dold for his implementation of Eppstein's efficient set reconciliation method. We thank Werner Koch for insightful discussions and for implementing ECDSA with RFC 6979 support in `libgcrypt`. We thank Krista Grothoff and Ralph Holz for editing the paper.

References

1. Essers, L.: German court finds domain registrar liable for torrent site's copyright infringement. http://www.itworld.com/print/403869 (February 2014)
2. European Parliament: Resolution on the EU-US Summit of 28 November 2011 (November 2011) P7-RC-2011-0577.
3. Anonymous: The collateral damage of internet censorship by dns injection. ACM SIGCOMM Comp. Comm. Review **42**(3) (July 2012) 22–27
4. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: DNS Security Introduction and Requirements. IETF RFC 4033 (Mar. 2005)
5. D. J. Bernstein: Dnscurve: Usable security for dns. http://dnscurve.org/ (August 2008)
6. Herzberg, A., Shulman, H.: Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org. In: CNS 2013. The Conference on Communications and Network Security. IEEE, IEEE (2013)
7. Soghoian, C., Stamm, S.: Certified lies: Detecting and defeating government interception attacks against SSL. In: Proc. 15th. Int. Conf. Financial Cryptography and Data Security. (Mar 2011)
8. Stiegler, M.: An introduction to petname systems. http://www.skyhunter.com/marcs/petnames/IntroPetNames.html (February 2005)
9. Wilcox-O'Hearn, Z.: Names: Decentralized, secure, human-meaningful: Choose two. http://zooko.com/distnames.html (Jan 2006)
10. Wachs, M., Schanzenbach, M., Grothoff, C.: On the feasibility of a censorship resistant decentralized name system. In: Foundations & Practice of Security. (2013)
11. Swartz, A.: Squaring the triangle: Secure, decentralized, human-readable names. http://www.aaronsw.com/weblog/squarezooko (January 2011)
12. Rivest, R.L., Lampson, B.: SDSI – a simple distributed security infrastructure. http://groups.csail.mit.edu/cis/sdsi.html (1996)
13. Bernstein, D.J.: Curve25519: new diffie-hellman speed records. In: In Public Key Cryptography (PKC), Springer-Verlag LNCS 3958. (2006)
14. Mockapetris, P.: Domain names - implementation and specification. RFC 1035 (Standard) (November 1987)
15. Thomson, S., Huitema, C., Ksinant, V., Souissi, M.: DNS Extensions to Support IP Version 6. RFC 3596 (Draft Standard) (October 2003)
16. Wachs, M.: A Secure Communication Infrastructure for Decentralized Networking Applications. PhD thesis, Technische Universität München (under submission)
17. Deccio, C., Sedayao, J., Kant, K., Mohapatra, P.: Quantifying dns namespace influence. Comput. Netw. **56**(2) (February 2012) 780–794
18. Evans, N., Grothoff, C.: R^5N: Randomized Recursive Routing for Restricted-Route Networks. In: 5th Int. Conf. on Network and System Security. (2011) 316–321
19. Ulrich, A., Holz, R., Hauck, P., Carle, G.: Investigating the OpenPGP Web of Trust. In: 16th European Symposium on Research in Computer Security (ESORICS). (September 2011)
20. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proc. 13th USENIX Security Symposium. (August 2004)
21. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf (2008)
22. http://dot-bit.org/: The Dot-BIT project, A decentralized, open DNS system based on the bitcoin technology. http://dot-bit.org/ (April 2013)
23. Ford, B.A.: UIA: A Global Connectivity Architecture for Mobile Personal Devices. PhD thesis, Massechusetts Institute of Technology (2008)

A Special Features

This appendix describes some additional special features in GNS that are used to deal with corner cases that a practical system needs to deal with, but that might only be relevant for a subset of the users.

A.1 Automatic Shortening

Once Dave's client translates "www.carol.+" to "www.carol.buddy.gnu", Dave can resolve "carol.buddy.gnu" to Alice's public key and then lookup the IP address for Alice's server under the respective key in the DHT. At this point, Dave's GNS system will also learn that Alice has set her "NICK" record to "carol". It will then check if the name "carol" is already taken in Dave's zone, and—if "carol" is free—offer Dave the opportunity to introduce a PKEY record into Dave's zone that would *shorten* "carol.buddy.gnu" to "carol.gnu".

Alternatively, the record could be automatically added to a special *shorten zone* that is, in addition to the master zone, under Dave's control. In this case, Alice would become available to Dave under "carol.shorten.gnu", thus highlighting that the name was created by automatic shortening within the domain name.

In either case, shortening eliminates Bob from the trust path for Dave's future interactions with Alice. Shortening is a variation of trust on first use (TOFU), as compromising Bob afterwards would no longer compromise Dave's path to Alice.

A.2 Relative Names in Record Values

GNS slightly modifies the rules for some existing record types in DNS. In particular, names in DNS values are always absolute; GNS allows the notation ".+" to indicate that a name is relative. For example, consider "CNAME" records in DNS, which map an alias (label) to a canonical name: as specified in RFC 1035 [14], the query can (and in GNS will) be restarted using the specified "canonical name". The difference between DNS and GNS is that in GNS, the canonical name can be a relative name (ending in ".+"), an absolute GNS name (ending in ".zkey") or a DNS name.

As with DNS, if there is a "CNAME" record for a label, no other records are allowed to exist for the same label in that zone. Relative names using the ".+" notation are not only legal in "CNAME" records, but in all records that can include names. This specifically includes "MX" and "SOA" records.

A.3 Dealing with Legacy Assumptions: Virtual Hosting and TLS

In order to integrate smoothly with DNS, GNS needs to accommodate some assumptions that current protocols make. We can address most of these with the "LEHO" resource record. In the following, we show how to do this for Web hosting. There are two common practices to address here; one is virtual hosting

(i. e. hosting multiple domains on the same IP address); the other is the practice of identifying TLS peers by their domain name when using X.509 certificates.

The problem we encounter is that GNS gives additional and varying names to an existing service. This breaks a fundamental assumption of these protocols, namely that they are only used with globally unique names. For example, a virtually hosted website may expect to see the HTTP header `Host: www.example.com`, and the HTTP server will fail to return the correct site if the browser sends `Host: www.example.gnu` instead. Similarly, the browser will expect the TLS certificate to contain the requested "www.example.gnu" domain name and reject a certificate for "www.example.com", as the domain name does not match the browser's expectations.

In GNS, each user is free to pick his own petname for the service. Hence, these problems cannot be solved by adding an additional alias to the HTTP server configuration or the TLS certificate. Our solution for this problem is to add the *le*gacy *ho*stname record type ("LEHO") for the name. This record type specifies that "www.example.gnu" is known in DNS as "www.example.com". A proxy between the browser and the web server (or a GNS-enabled browser) can then use the name from this record in the HTTP `Host:` header. Naturally, this is only a legacy issue, as a new HTTP header with a label and a zone key could also be introduced to address the virtual hosting problem. The LEHO records can also be used for TLS validation by relating GNS names to globally unique DNS names that are supported by the traditional X.509 PKI. Furthermore, GNS also supports TLSA records, and thus using TLSA records instead of CAs would be a better alternative once browsers support it.

A.4 Handling TLSA and SRV records

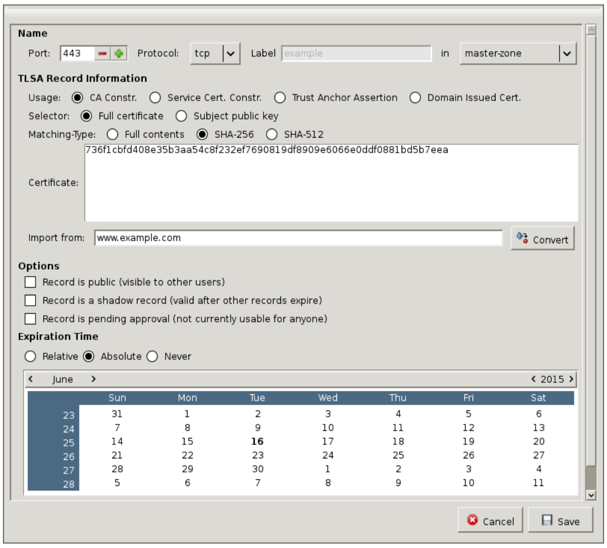
TLSA records are of particular interested for GNS, as they allow TLS applications to use DNSSEC as an alternative to the X.509 CA PKI. With TLSA support in GNS, GNS provides an alternative to X.509 CAs and DNSSEC using this established standard. Furthermore, GNS does not suffer from the lack of end-to-end verification that currently plagues DNSSEC.

However, to support TLSA in GNS a peculiar hurdle needs to be resolved. In DNS, both TLSA and SRV records are special in that their domain names are used to encode the service and protocol to which the record applies. For example, a TLSA record for HTTPS (port 443) on `www.example.com` would be stored under the domain name `_443._tcp.www.example.com`.

In GNS, this would be a problem since dots in GNS domain names are supposed to always correspond to delegations to another zone. Furthermore, even if a special rule were applied for labels starting with underscores, this would mean that say the A record for `www.example.com` would be stored under a different key in the DHT than the corresponding TLSA record. As a result, an application would experience an unpredictable delay between receiving the A record and the TLSA record. As a TLSA record is not guaranteed to exist, this would make it difficult for the application to decide between delaying in hope of using a TLSA record (which may not exist) and using traditional X.509 CAs for authentication (which may not be desired and likely less secure).

GNS solves this problem by introducing another record type, the BOX record. A BOX record contains a 16-bit port, a 16-bit protocol identifier, a 32-bit embedded record type (so far always SRV or TLSA) and the embedded record value. This way, BOX records can be stored directly under `www.example.com` and the corresponding SRV or TLSA values are thus never delayed — not to mention the number of DHT lookups is reduced. When GNS is asked to return SRV or TLSA records via DNS, GNS recognizes the special domain name structure, resolves the BOX record and automatically unboxes the BOX record during the resolution process. Thus, in combination with the user interface (Figure 2) GNS effectively hides the existence of BOX records from DNS users.

We note that DNS avoids the problem of indefinite latency by being able to return NXDOMAIN in case a SRV or TLSA record does not exist. However, in GNS NXDOMAIN is not possible, largely due to GNS's provisions for query privacy. Furthermore, DNS can solve the efficiency problem of a second lookup by using its "additional records" feature in the reply. Here, a DNS server can return additional records that it believes may be useful but that were not explicitly requested. However, returning such additional records might not always work, as DNS implementations can encounter problems with the serious size restrictions (often just 512 bytes) on DNS packets. As GNS replies can contain up to 63 kB of payload data, we do not anticipate problems with the size limit in GNS even for a relatively large number of unusually big TLSA records.



Name

Port: 443 Protocol: tcp Label example in master-zone

TLSA Record Information

Usage: ● CA Constr. ○ Service Cert. Constr. ○ Trust Anchor Assertion ○ Domain Issued Cert.

Selector: ● Full certificate ○ Subject public key

Matching-Type: ○ Full contents ● SHA-256 ○ SHA-512

Certificate: 736f1cbfd408e35b3aa54c8f232ef7690819df8909e6066e0ddf0881bd5b7eea

Import from: www.example.com

Options

□ Record is public (visible to other users)

□ Record is a shadow record (valid after other records expire)

□ Record is pending approval (not currently usable for anyone)

Expiration Time

○ Relative ● Absolute ○ Never

‹ June › ‹ 2015 ›

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
23	31	1	2	3	4	5	6
24	7	8	9	10	11	12	13
25	14	15	**16**	17	18	19	20
26	21	22	23	24	25	26	27
27	28	29	30	1	2	3	4
28	5	6	7	8	9	10	11

Fig. 2. The user can remain unaware of the behind-the-scenes boxing when creating TLSA records in the GNS zone management interface.

A.5 Revocation

In case a zone's private key gets lost or compromised, it is important that the key can be revoked. Whenever a user decides to revoke a zone key, other users must be notified about the revocation. However, we cannot expect users to explicitly query to check if a key has been revoked, as this increases their latency (especially as reliably locating revocations may require a large timeout) and bandwidth consumption for every zone access just to guard against the relatively rare event of a revoked key. Furthermore, issuing a query for zone revocations would create the privacy issue of revealing that a user is interested in a particular zone. Existing methods for revocation checks using certificate revocation lists in X.509 have similar disadvantages in terms of bandwidth, latency increase and privacy.

Instead of these traditional methods, GNS takes advantage of the P2P overlay below the DHT to distribute revocation information by flooding the network. When a peer wants to publish a revocation notice, it simply forwards it to all neighbors; all peers do the same when the receive previously unknown valid revocation notices. However, this simple-yet-Byzantine fault-tolerant algorithm for flooding in the P2P overlay could be used for denial of service attacks. Thus, to ensure that peers cannot abuse this mechanism, GNS requires that revocations include a revocation-specific proof of work. As revocations are expected to be rare special events, it is acceptable to require an expensive computation by the initiator. After that, all peers in the network will remember the revocation forever (revocations are a few bytes, thus there should not be an issue with storage).

In the case of peers joining the network or a fragmented overlay reconnecting, revocations need to be exchanged between the previously separated parts of the network to ensure that all peers have the complete revocation list. This can be done using bandwidth proportional to the difference in the revocation sets known to the respective peers using Eppstein's efficient set reconciliation method. In effect, the bandwidth consumption for healing network partitions or joining peers will then be almost the same as if the peers had always been part of the network.

This revocation mechanism is rather hard to disrupt for an adversary. The adversary would have to be able to block the flood traffic on all paths between the victim and the origin of the revocation. Thus, our revocation mechanism is not only decentralized and privacy-preserving, but also much more robust compared to standard practice in the X.509 PKI today, where blocking of access to certificate revocation lists is an easy way for an adversary to render revocations ineffective. This has forced vendors to include lists of revoked certificates with software updates.

A.6 Shadow Records

GNS records can be marked as "shadow records"; the receiver only interprets shadow records if all other records of the respective type have expired. This is useful to ensure that upon the timeout of one set of records the next set of records is immediately available. This may be important, as propagation delays in the DHT are expected to be larger than those in the DNS hierarchy.