On the Feasibility of a Technological Response to the Surveillance Morass

Joan Feigenbaum and Jérémie Koenig

Yale University, New Haven CT, 06520-8285 USA
E-mail: {joan.feigenbaum, jeremie.koenig}@yale.edu

Abstract. We consider mass surveillance from a computer-science perspective. After presenting some objections to the behavior of the US National Security Agency and its counterparts in allied nations (emphasizing technical problems associated with such behavior, rather than political, legal, and social problems), we propose a grass-roots, technological response: decentralized cloud services, facilitated by open-source, decentralized configuration-management tools.

1 Introduction

Since June 2013, information leaked by Edward Snowden has revealed that the US National Security Agency (NSA) has for years been conducting dragnet surveillance both domestically and internationally, covertly sabotaging security standards and products, and pressuring major US technology companies to cooperate in its activities. Apparently, sister agencies in allied nations (particularly Canada, the UK, Australia, and New Zealand – the four other Anglophone nations of the "five eyes" consortium) have collaborated with NSA in the collection, storage, and mining of unprecedented amounts of sensitive information. In other words, "all our paranoid dreams of the past twenty years have come true."[1]

In this paper, we consider the surveillance morass from a computer-science perspective. In Section 2, we present some objections to government surveillance of entire populations, emphasizing the technical problems associated with it rather than the political, legal, and social problems. In the same section, we ask whether the US business community has both the incentive and the power to bring about change in its government's policies on mass surveillance and security sabotage; we conclude that there are some reasons for hope but also some reasons for despair on that front. In Sections 3 and 4, we outline a possible grass-roots, technological response to our current predicament, to wit: a transition to more decentralized cloud services, facilitated by open-source, decentralized configuration-management tools.

[1] Call for Papers, Cambridge Security Protocols Workshop, 2014

2 The Surveillance Morass

2.1 Problem Description

We use the term "surveillance morass" to refer both to intelligence-agency practices that we find objectionable and to ambient conditions in the Internet that enable these practices and may make it difficult to put a stop to them.

Objectionable practices have been covered steadily by the news media since the Snowden story broke in June 2013. Indeed, as reported in [1], the NSA itself provided the following summary of its "collection posture" in a slide presentation at a multinational meeting of intelligence agencies in 2011:

> Collect it all. Process it all. Exploit it all. Partner it all. Sniff it all. Know it all.

To accomplish its panoptic goal, the agency, often in cooperation with its sister agencies in allied nations, has been collecting massive amounts of communications "metadata" (including but not limited to the phone numbers, date, time, location, and duration of every cell-phone call made in the US), surveilling both corporate databases and user-generated data of major Internet companies (with the companies' cooperation when it can be obtained and by breaking into data centers when it cannot), and engaging in security sabotage (covert and deliberate undermining of cryptographic standards and products and of the standardization process). A comprehensive explanation of these practices is beyond the scope of this paper and has been undertaken by others; see, for example, Greenwald's recent book [1].

An essential enabler of this breathtaking surveillance regime is the ubiquity of computers, smart phones, and communication networks in everyday life. More and more of our daily activities in commerce, education, government, recreation, and even friendship and romance are mediated by electronic devices that create records of these activities, either as their primary products or as by-products. A growing number of ad-supported cloud services require companies to retain, interpret, and mine records of our daily activities so that ads can be targeted well enough to fetch high prices. Arguments against targeted advertising and the data mining that supports it have been advanced for years, but their abolition could spell the end of the Web as we know it [2]. The troves of personal data created by modern communication networks and cloud services are and will always be irresistible, fat targets for intelligence services.

To explain one of our technical objections to intelligence-agency overreach, we first recall a legal objection that many (but not all) participants in the debate have raised, *i.e.*, that dragnet surveillance is a *prima facie* violation of the Fourth Amendment of the US Constitution. Recall that the amendment guarantees that

> The right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures, shall not be violated, and no warrants shall issue, but upon probable cause, supported by oath or affirmation, and particularly describing the place to be searched, and the persons or things to be seized.

Unlike the First, Second, and Fifth Amendments, the Fourth has not traditionally played much of a role in American popular culture, but it played a huge role in American history. Rejection of "general warrants," under which agents of the British government subjected entire communities to search and seizure, was one the main reasons that 18th-century American colonists fought the Revolutionary War.

The Fourth Amendment is an early expression of a consistent theme in discussions of law enforcement and intelligence generally and of electronic surveillance in particular: the belief that privacy and security are both important goals but that they are inherently at odds with each other. One straightforward interpretation of the amendment is that, under normal circumstances, citizens are entitled to personal privacy but that, if there is credible and particularized suspicion that a specific citizen has committed a specific crime (thereby violating others' security), government authorities may be granted a warrant to search his home and seize his possessions (thereby violating his privacy). This simple example of the need for law-enforcement agencies to "balance" or "trade off" privacy and security makes sense intuitively, and US citizens have centuries of experience with lawfully obtained search warrants' enabling police officers to catch criminals and collect evidence that can be used to convict them in court.

In the debate about NSA surveillance, however, some people have implicitly made a much stronger and more general assumption that is *not* supported by real-world experience or by scientific research, *i.e.*, that there is a robust, tunable tradeoff between security and privacy in which the security of society at large is always guaranteed to improve if the privacy of individuals within the society is allowed to erode. Even some civil-liberties supporters who argue that limits must be placed on NSA data collection say things like "it's a tradeoff. If we had perfect information, then we'd have perfect security, but we cannot tolerate the level of government intrusion necessary to achieve perfect information." *Nothing in the scientific literature establishes the existence of this type of robust, tunable tradeoff.* The fact that some effective security measures, *e.g.*, lawfully authorized search and seizure, cause some loss of privacy does *not* imply that loss of privacy *per se* causes or is even positively correlated with increased security.

2.2 Threats Posed by Personal-Data Collection on a Massive Scale

It is entirely possible that storage and mining of personal data on the scale implied by the NSA's collection posture is inherently insecure and destined to cause the nation more harm than good.

One threat clearly posed by the mere existence of personal-data hoards of unprecedented size is mission creep. Just as cloud-service providers' treasure troves of personal data proved too tempting for intelligence agencies to resist, intelligence agencies' troves are, in our opinion, likely to be used for purposes other than intelligence. Indeed, there have already been reports of diverse organizations' requesting access to NSA data in order to thwart drug trafficking, cyber attacks, money laundering, counterfeiting, and copyright infringement [3]; as of August 2013, the NSA claimed to have turned down all of those requests,

but will it resist forever, no matter what it is offered in return? On a more banal note, there have been reports of NSA employees' abusing their access to surveillance data in order to spy on their romantic partners and ex-partners [4]; the practice is known as LOVEINT, by analogy with SIGINT (signals intelligence) and HUMINT (human intelligence).

A second clear threat is infiltration and corruption of data hoards. Given that the NSA has itself infiltrated data centers of Google and Yahoo! [5], the agency would be foolish to assume that no one could infiltrate its data centers. Infiltrators need not act dramatically and quickly, *e.g.*, by appropriating large amount of money using stolen banking credentials (although that would be destructive enough). They could, for example, alter data in critical but subtle ways that are hard to detect, particularly since most of the data in these hoards will be accessed rarely if ever.

Although dragnet collection of sensitive data poses substantial threats, whether it provides substantial value remains unclear. A presidential review group convened to study the NSA controversy found no evidence that universal collection of cell-phone metadata contributed useful information that could not have been obtained using conventional intelligence-gathering techniques [6]. Even sensible uses of cell-phone calling records by intelligence agencies are apparently carried out in a more privacy-invasive manner than they need be. For example, the NSA's "co-traveler" program [7] finds unknown associates of known (presumably legitimate) surveillance targets by first intersecting cell-tower dumps from times and locations at which a particular known target appeared and then interpreting the intersection as the set of cell-phone numbers of people who may be "traveling with" the known target. By using *privacy-preserving set intersection*, a well studied cryptographic problem for which there are efficient solutions [8–10], the agency could arrive at the same (small) set of co-travelers' phone numbers *without* learning the phone numbers of the (large) set of innocent people who happen to have used one of the same cell towers at a relevant time. No doubt there are other well understood protocols in the vast cryptographic literature that could be used to find truly useful intelligence without revealing massive amounts of private information about ordinary citizens.

The claim that dragnet surveillance is acceptable when "metadata," rather than "data," are all that is gathered is highly dubious. Technically, there is simply no well defined distinction between metadata and data: One program's metadata are another program's data; for example, from an email client's point of view, sender's and receiver's IP addresses may be metadata, but, from a router's point of view, they are data. Socially, the claim that "who, when, where, and for how long" information about a person's cell-phone calls (aka metadata) is less revealing or less deserving of privacy protection than the content of his calls (aka data) does not pass the laugh test. Clearly, there are situations in which all of the metadata are pretty well known to the authorities anyway, and the interesting question is what the people on the phones are saying; there are just as many situations, however, in which the questions of interest are precisely "with whom, when, where, and for how long is this person communicating?"

Government agencies should not obtain answers to any of these questions without particularized suspicion.

The claim that communications have only been "intercepted" or that data have only been "seized" when a human being hears, reads, or otherwise consumes them ignores the reality of the Big-Data era we live in. The shift from human-mediated to computer-mediated surveillance does not make mass surveillance less objectionable. It may even make it more so, because imperfect, probabilistic algorithms now interpret people's words and activities on behalf of government agencies with enormous power (not just on behalf of companies that want to target ads).

2.3 Security Sabotage

Recall that we use the term "security sabotage" to refer to government agencies' covert and deliberate weakening of crypto and security standards and products through interference in the work of standards bodies or companies. Sabotage is one, but by no means the only, approach taken in the NSA's Bullrun program [11], the goal of which is to "defeat the encryption used in specific network communication technologies."

For security sabotage to be effective as a tool of intelligence and law enforcement, the weaknesses inserted into standards and products must be usable by intelligence and law-enforcement agents but not by the very terrorists and criminals that they are intended to defeat. *There is no reason to believe that this is the case.* On the contrary, security sabotage has backfired before, *e.g.*, in the case of the mobile-phone system built by Vodafone Greece [12]. The system was intended for use by members of the Greek government and senior civil servants; it contained "built-in wiretapping facilities" for official use. Hackers subverted these facilities and managed to eavesdrop on the Prime Minister, the Mayor of Athens, and many other high-level officials.

Crypto and security researchers have worked for decades, often at taxpayer expense, to create the mathematical and technological foundation for a secure information environment. Security sabotage is tantamount to betrayal of those researchers and the taxpayers who support them and to vandalism of that foundation. It is not only unethical and heavy-handed but potentially economically destructive; to remain dominant, the US tech industry will require customers' trust, and that trust has been violated.

Finally, we believe that security sabotage invites bad product design and implementation. If inventors and developers believe that standard cryptographic protocols are likely to have hidden features that enable government eavesdropping, they may opt to use nonstandard, inadequately vetted protocols or even attempt to design their own. Cryptography and security are difficult, highly specialized areas in which expert evaluation and standardization processes have developed over decades (and are still developing). This painstaking and expensive development effort will have been wasted if the resulting processes are perceived to have been corrupted by government surveillance agencies.

2.4 The US Business Community

As described by Schneier [13], pervasive use of cloud computing has given rise to a regime of Internet use that is reminiscent of feudalism. By entrusting all of our personal data and the records of all of our online activity to one (or a very small number of) for-profit cloud-service providers (Google, Facebook, Yahoo!, *etc.*), users play the role of feudal peasants; we are dependent on these providers and must be loyal to them or endure significant switching costs. Similarly, the providers play the role of feudal lords in that they command our loyalty and profit from it, but they are to some extent obligated to treat us decently, because we could abandon one of them for another, and the most talented and entrepreneurial of us could even rise up and overtake them. Other technology critics have explored the feudal metaphor, most notably Lanier [14], who calls the providers "Lords of the Cloud" and argues that their business models are destroying the world economy.

Unsurprisingly, the Lords of the Cloud are unhappy about NSA's surveilling their users, breaking into their data centers, implying to journalists that they have willingly cooperated with NSA's data-collection programs, and refusing to allow them to clarify the extent to which they actually have cooperated (in the sense that they have responded to subpoenas and National Security Letters, details of which are usually classified). Their CEOs have met with President Obama to express their unhappiness, and eight major firms have issued a joint objection to the current surveillance regime, together with five principles that could inform a better regime [15].

We applaud the Lords for this action and think that their proposed principles are reasonable. Moreover, we recognize that business lobbies can have enormous influence on US electoral politics and congressional legislation; tech-industry support for anti-surveillance candidates and legislative efforts would be welcome. Unfortunately, we see at least two reasons that such efforts cannot be expected to lead to significant change in the near future. Although the tech industry is rich and powerful, it is not nearly as powerful in Washington DC as the military and intelligence communities. Furthermore, the Lords of the Cloud have limited credibility in opposing surveillance. At the core of their business models is the exploitation of personal information for the purpose of targeting ads, and, as explained in Section 2.1, their collection of that personal information is a key component of the surveillance morass.

Of course, tech is not the only business sector in the US, and corporations in general probably do not like the extent to which they and their customers are beholden to the Lords of the Cloud. Perhaps they will lend their support to the vision outlined in Section 4 below, just as they lent their support 15 years ago to open-source development of webservers and other Web 1.0 components.

3 The Still Somewhat Decentralized Internet

As explained in Section 2.1, one major enabler of mass surveillance is the popularity of ad-supported cloud services. A crucial feature of these services is that

they are *centralized*. We use this term to describe a service or system that is controlled by one principal; note that centralized services may be *distributed*, *i.e.*, they may be executed on multiple machines. By contrast, *decentralized* services or systems are not only executed on multiple machines but controlled by multiple principals that may have little or no trust in each other. In this section, we identify some of the factors underlying this trend toward centralization.

By managing Internet services for a large number of users, the Lords of the Cloud are able to realize huge economies of scale and provide services that are "free" in that end users are not charged directly. Because their revenue model revolves around the exploitation of user data, their incentive is to centralize those data on their servers. The costs associated with this centralization, namely the loss of privacy and the ease of mass surveillance, are borne mainly by society as a whole rather than by individual users of the Lords' services. They are treated by cloud-service providers as externalities.

We argue that the utility provided by the Lords of the Cloud resides primarily in their production of a particular kind of software. We look to the open-source movement for an alternative regime for the production of this software that can better account for the overall public interest.

3.1 The Internet's Decentralized Roots

The core infrastructure of the Internet was built to support a decent amount of decentralization. The Domain Name System (DNS) provides a decentralized global namespace in which administration of subtrees can be delegated. Other services can then take advantage of this infrastructure: Domain names can be embedded in the names of protocol-specific resources and used to resolve the hosts that provide the corresponding services.

For instance, in order to deliver an email message, a Mail Transport Agent (MTA) first extracts the domain part of the recipient's address. The MTA then performs a DNS request and obtains the Mail Exchanger (MX) records associated with the domain. These records specify a set of servers capable of receiving email for the domain, as well as associated priorities. The MTA can then try to contact these servers in order of priority until mail delivery succeeds. Because many other Internet services follow similar patterns, the procedure has been generalized across protocols and unified in the DNS Service (SRV) resource records [16],

Email and web are the most popular Internet applications built in this way, but many others exist. For example, the Andrew File System (AFS) is a distributed file system commonly used in large infrastructures. It relies on the Kerberos authentication protocol, which is often used in conjunction with user-account data published using the Lightweight Directory Access Protocol (LDAP). These protocols include a notion of independent administrative *realms* that are able to interoperate without prior trust. They can all function as global protocols by using DNS to record the servers associated with a realm.

Many of these decentralized technologies have been available since the 1980s, and some of them have been widely adopted in the context of information sys-

tems for medium-sized and large businesses. Indeed, LDAP and Kerberos form the basis of Active Directory, the service used on Microsoft Windows for sharing account information and authentication over the network. Nevertheless, despite great potential, none of them has become ubiquitous as an Internet protocol to the same extent that email and web protocols have, and Internet users tend to rely on centralized solutions instead.

3.2 The Lords' Economies of Scale

That a small number of companies dominate mass-market cloud services is unsurprising given the massive economies of scale that centralization enables. From the point of view of an isolated user, the services provided by the Lords of the Cloud are essentially free, because the Lords' marginal costs are essentially nonexistent. Alternatives are cost-prohibitive: Even for open, decentralized services such as email, small-scale providers incur significant costs per user, *e.g.*, for hardware and labor.

Large service providers maintain infrastructures in which every part of the administration process is automated. At very large scale, system administration is essentially software development, where the "machine" that runs the software in question is the whole infrastructure rather than a single computer. As in all software development, there are huge initial costs, and a highly skilled workforce is required; however, the marginal cost per user is essentially zero.

Like traditional proprietary-software companies, the Lords of the Cloud develop infrastructure software that addresses users' needs only as a secondary objective, to the extent that such development efforts enhance the companies' profits. As a consequence, infrastructure-software development is focused on centralized solutions in which users' data are collected by the service provider, rather than remaining under users' control. Although it has been pointed out repeatedly that these solutions come at the expense of users' privacy, they are sustainable: Many of us do not suffer directly from loss of privacy to a faceless corporation or even to the surveillance state; more importantly, one person's choice of provider has very little impact even on his own vulnerability, much less on prevailing forms of cloud-service architecture.

Rather, centralization has problematic consequences primarily in the aggregate, when a large percentage of the world's email transits through a few companies' servers, and most electronic communications on the planet are made available to at least one country's espionage agencies and their associates. These consequences do not factor into the economic calculations of a service provider, its users, or its paying customers (most of whom are advertisers).

3.3 Open-Source Software as a Model

A development regime for infrastructure software that prioritized users' interests would look very different. To an extent, such a regime can be found in the open-source community, where users of the software, among them many corporations,

pool their resources and collaborate directly on its development (instead of pooling their resources indirectly by paying the developer of a proprietary product). Because the developers are a subset of the intended users, open-source software is constructed with a very different agenda.

It is not entirely clear how the open-source regime of collaborative development can be applied to infrastructure software. However, one thing is quite clear: Decentralization becomes a requirement; participants can trust the *code* that they exchange, because of the transparency and traceability of the development process, but that does not mean that they can trust each other to share *administrative privileges* over a common Internet infrastructure. Fortunately, as discussed in Section 3.1, there are many protocols for decentralized systems that could be used as components, including some Internet protocols that are already widely deployed.

As costs of both hardware and bandwidth have continued to decline, self hosting has become more and more affordable. In the US and other wealthy countries, always-on Internet connections are very common. One can purchase small plug computers the size of a home router for \$50 – \$100. Alternatively, one can rent a virtual machine in a data center for a few dollars per month. Still, the skills and time required to operate servers of any kind remain obstacles to widespread use of self hosting. This has prompted several groups to develop home-server operating systems targeting plug computers with enhanced privacy and control as a stated goal.

In 2010, Eben Moglen gave a series of talks in which he invited the open-source community to create the *Freedom Box*: a plug computer packed with ready-to-use privacy-enhancing software [17]. More recently, the arkOS project [18] has focused on self hosting of email and web content. In both of those cases, the model is that of a home-network appliance: a small box with a limited purpose and few parameters that can be tuned using a convenient web interface. Here "infrastructure as software" applies literally: A general-purpose operating system is pre-configured to fulfill a certain specialized role; very little configuration remains to be done by the end-user, and project proponents hope that this will facilitate deployment.

Projects of this kind are interesting, and appliance-style servers may have an important role to play in a shift towards decentralized cloud services. However, the network-appliance paradigm by itself cannot provide service comparable to that provided by the Lords of the Cloud; the Lords' level of availability and reliability can only be achieved through redundancy, and one needs a mechanism by which plug servers can cooperate to provide it. Furthermore, Freedom Box and arkOS target only very small institutions, *e.g.*, households and perhaps small businesses; in order to replace the Feudal Internet with a more open and democratic Renaissance Internet, we will need to enlist the participation of many sorts of institutions, including large ones.

We believe that *decentralized configuration management* can address these limitations by bridging the gap between decentralized, but low-level, Internet

protocols and well established, but centralized, system-administration methods that are typically used for large infrastructures.

4 Decentralized Configuration Management

While many Internet protocols are designed to operate in a decentralized manner, the methods and software currently used by the participants to deploy them typically assume centralized control over the infrastructure. By relaxing this assumption, we can build an open-source framework that would allow groups of people to cooperate on the provision of the same kinds of services that are now provided by the Lords of the Cloud in a centralized, albeit distributed, fashion.

4.1 Configuration-Management Systems

The theme of "system administration as software development" is of course not exclusive to the very large-scale infrastructures of cloud-service providers; it can be useful for smaller deployments as well. Systems for centralizing the configuration and administration of medium-sized and large infrastructures are usually known as *configuration-management systems*.

Typically, the configuration for the whole infrastructure is stored *as code* in a central repository. The configuration is usually placed under revision control (as is done for more typical software-development tasks). The configuration-management system provides mechanisms to propagate the configuration from the central repository to each client machine in a reliable and repeatable way.

In principle, because a configuration-management system captures the state of an infrastructure as code, it should be possible to use it for collaborative system administration. Indeed, in a typical enterprise setup, a team of administrators shares access to the configuration repository, perhaps following a process similar to those used by teams of programmers. In a hypothetical collaborative cloud service, participants would pool their resources and set up their machines to use a common configuration repository, collaborating and "contributing" to this shared configuration so as to accommodate their common needs in a mutually agreeable fashion.

Unfortunately, this model cannot scale. In this arrangement, the participants must trust each other. In most cases, granting a set of participants access to the configuration repository is tantamount to giving them full privileges on each others' machines. Although it is sometimes possible to set up fine-grained control of access to the configuration repository, every configuration-management system that we are aware of relies on a centralized database in which the configuration of the whole infrastructure is stored. The integrity of this database has to be trusted by all participants in an all-or-nothing fashion.

4.2 Decentralized Configuration Management

Truly *decentralized configuration management* would allow two or more independent participants to express configuration *policies* in a repository that they alone

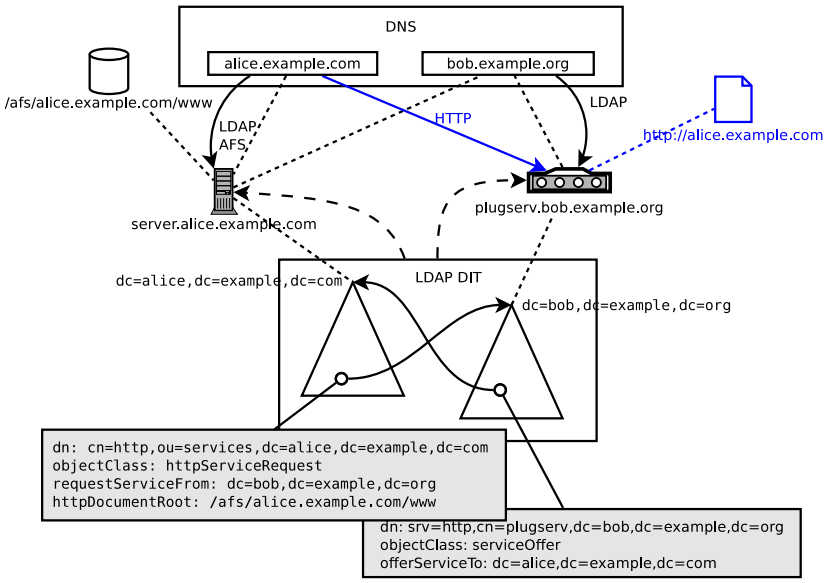

Fig. 1. Example scenario for the envisioned configuration-management framework. Dotted lines denote the associations between servers and the services they provide, solid lines denote references, and dashed lines show the flow of configuration information.

control. Such policies may specify that some forms of collaboration are desired or permissible. The host-configuration mechanism of a given machine could then access the repositories of several participants in addition to that of its owner and derive a configuration for that machine that satisfies all of the policies.

As a concrete example of such a system, consider the situation depicted in Fig. 1. Alice and Bob own `alice.example.com` and `bob.example.org`. Because they control the contents of these DNS zones, they can designate LDAP servers for the subtrees rooted at `dc=alice`, `dc=example`, `dc=com` and `dc=bob`, `dc=example`, `dc=org` in the global Directory Information Tree (DIT). Therefore, they also control the contents of these subtrees, which they use to hold their configuration policies.

Alice's machine `server.alice.example.com` hosts her DNS zone and the corresponding DIT, as well as the AFS volume `www`, where the data files for her website reside. Bob's server `plugserv.bob.example.org` likewise hosts his DNS zone and LDAP DIT. Now suppose that Alice and Bob are to cooperate on hosting the website `http://alice.example.com`; Bob's machine is to serve the static contents of Alice's website, which it can find in `/afs/alice.example.com/www`. For this to happen,

- Alice has to trust Bob to provide the service faithfully;
- Bob must to be willing to host Alice's website on his server.

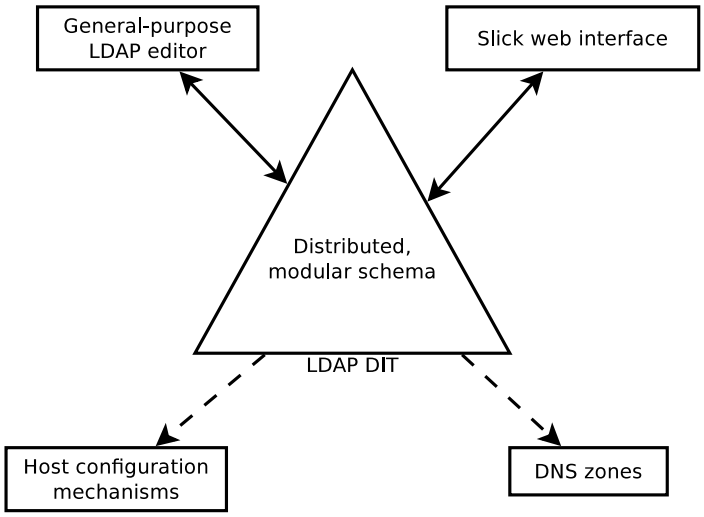

Fig. 2. Several possible components of the framework and their relationships

If these two conditions were met, Alice would modify her DNS zone in such a manner that `alice.example.com` would point to Bob's `plugserv` as the associated HTTP server, and Bob would configure that machine to serve the appropriate content.

Our goal is to automate this process so that Alice and Bob can express their preferences in a common configuration repository in the form of *service requests* and *service offers*. These requests and offers could then be matched to one another so as to compute the configuration of each individual host.

4.3 Overall Structure

Figure 2 shows the overall structure of the system we envision. An LDAP directory is the obvious candidate to hold the configuration policy, because it is possible to delegate maintenance and control of subtrees in the directory to different administrative principals. Ultimately, such a system would provide a user interface to assist the administrator in creating this configuration policy. However, in the prototyping phase, a general purpose LDAP editor would be enough. It could also serve as an escape hatch for advanced users who wish to access parameters unavailable through user interfaces or to introduce their own.

Host-configuration software can then compute the configuration of each individual machine using data from different participants. Assuming the LDAP schema has been carefully defined, it would be possible for different implementations of such host-configuration mechanisms to coexist. In Fig. 1, the machines `server.alice.example.com` and `plugserv.bob.example.org` might be running completely different operating systems. The computed configuration can

include any DNS zones that have to be published. In the case of Fig. 1, `bob.info` is served by both Alice's `server` and Bob's `plugserv`. In fact, the two servers can independently compute the zone's contents from the data in the repositories, eliminating the need for DNS zone transfer.

Furthermore, these configuration mechanisms could use substantially different approaches. At one end of the spectrum, a human adminstrator could inspect the contents of the directory manually and carefully configure a given machine to match its role in the infrastructure. While this approach would ultimately defeat the purpose of automating configuration, it could be useful as a stop-gap measure when prototyping new schema components. As an example of the other extreme, the data from the directory could be used to specialize and compile a complete operating system image, possibly of the kind proposed in [19]. Most likely, the usual case would comprise some kind of automatic host-configuration tool operating on a conventional operating system. This could be achieved by a new tool, a preliminary version of which we have prototyped and are experimenting with.[2] Alternatively, one could set out to modify or specialize an existing configuration-management system to integrate similar capabilities.

5 Conclusions and Future Directions

After exploring the surveillance morass from a computer-science perspective, we have concluded that centralized cloud services play a crucial role in perpetuating the current, objectionable state of affairs. We have proposed an alternative approach to the construction of global-scale cloud services, based on open-source, decentralized configuration-management tools.

Research on the question of how simultaneously to provide scalable cloud services, user privacy, and support for *lawful* surveillance is fairly new, and open questions abound. We give just two of them here.

As we have argued in Section 2.2, collection and storage of massive numbers of phone-call records and other communications "metadata" are potentially harmful and may not even be necessary for effective pursuit of criminals and terrorists. In at least some realistic use cases, well studied cryptographic techniques, such as privacy-preserving set intersection, can be used to identify and track suspects while *not* identifying or tracking innocent bystanders or any other non-suspects. There may, however, be inherent limitations to this approach. In order to use most of the relevant techniques in the literature, one must start with a well defined function that one wants to compute and then design and implement a protocol that computes it in a privacy-preserving manner. An intelligence agency, however, may not know in advance exactly what it wants to compute. For example, it may uncover information that appears relevant to an investigation and suggests other sources of potentially relevant information but does not by itself suggest a well defined function to compute in a privacy-preserving manner; indeed, it may be precisely by collecting and examining more sensitive information

[2] `https://github.com/jeremie-koenig/ldapmin`

from the suggested sources that the investigators figure out what they need to compute.

Clearly, the technical approach put forth in Section 4 must be fully fleshed out before it can seriously challenge the centralized cloud-service regime that prevails today. Because a successful challenge would require the buy-in of large organizations, probably including for-profit corporations, analysis of incentives and other economic aspects of our proposal is necessary along with technical development.

References

1. Greenwald, G.: No Place to Hide: Edward Snowden, the NSA, and the U. S. Surveillance State, Metropolitan Books, New York (2014)
2. Goldfarb, A., Tucker, C.E.: Online Advertising, Behavioral Targeting, and Privacy. Communications of the ACM. 54(5), 25–27 (2011)
3. Lichtblau, E., Schmidt, M.S.: Other Agencies Clamor for Data NSA Compiles. The New York Times (Aug. 3, 2013)
4. Gorman, S.: NSA Officers Spy on Love Interests. The Wall Street Journal (Aug. 23, 2013)
5. Gellman, B., Soltani, A.: NSA Infiltrates Links to Yahoo, Google Data Centers Worldwide, Snowden Documents Say. The Washington Post (Oct. 30, 2013)
6. Clarke, R.A., Morell, M.J., Stone, G.R., Sunstein, C.R., Swire, P.: Liberty and Security in a Changing World, `http://www.scribd.com/doc/192387819/NSA-review-board-s-report` (Dec. 12, 2013)
7. Soltani, A., Gellman, B.: New Documents Show How the NSA Infers Relationships Based on Mobile Location Data. The Washington Post (Dec. 10, 2013)
8. Freedman, M. J., Nissim, K., Pinkas, B.: Efficient Private Matching and Set Intersection. In: Cachin, C., Camenisch, J. (eds.) Eurocrypt 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
9. Kissner, L., Song, D.: Privacy-Preserving Set Operations. In: Shoup, V. (ed.): Crypto 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
10. Vaidya, J., Clifton, C.: Secure set intersection cardinality with application to association rule mining. Journal of Computer Security. 13(4), 593–622 (2005)
11. The Guardian: Project Bullrun – classification guide to the NSA's decryption program, `http://www.theguardian.com/world/interactive/2013/sep/05/nsa-project-bullrun-classification-guide` (Sept. 5, 2013)
12. Prevelakis, V., Spinellis, D.: The Athens Affair. IEEE Spectrum. 44(7), 26–33 (2007)
13. Schneier, B.: When It Comes to Security, We're Back to Feudalism, `https://www.schneier.com/essay-406.html` (Nov. 26, 2012)
14. Lanier, J.: Who Owns the Future?. Simon and Schuster, New York (2013)
15. AOL, Apple, Dropbox, Facebook, Google, LinkedIn, Microsoft, Twitter, Yahoo!: Global Government Surveillance Reform, `http://reformgovernmentsurveillance.com` (2013)
16. Gulbrandsen, A., Vixie, P., Esibov, L.: A DNS RR for specifying the location of services (DNS SRV). RFC 2782 (Proposed Standard), `http://www.ietf.org/rfc/rfc2782.txt` (Feb. 2000). Updated in RFC 6335

17. Vaughan-Nichols, S.J.: Freedom box: Freeing the Internet one Server at a time, http://www.zdnet.com/blog/networking/freedom-box-freeing-the-internet-one-server-at-a-time/698 (Feb. 16, 2011)
18. Henderson, N.: Open Source Project arkOS Brings Simplicity to Self-Hosting, http://www.thewhir.com/web-hosting-news/open-source-project-arkos-brings-simplicity-to-self-hosting (Nov. 12, 2013)
19. Madhavapeddy, A., Mortier, R., Rotsos, C., Scott, D., Singh, B., Gazagnaire, T., Smith, S., Hand, S., Crowcroft, J.: Unikernels: Library Operating Systems for the Cloud. In: 18th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 461–472. ACM Press, New York (2013)