

Must Social Networking Conflict with Privacy?

Jonathan Anderson and Frank Stajano | University of Cambridge

Online social networks (OSNs) have serious privacy drawbacks, some of which stem from the business model. Must this be? Is the current OSN business model the only viable one? Or can we construct alternatives that are technically and economically feasible?

Conventional thinking is that online social network (OSN) users must be willing to give up on privacy. This view is often grounded in the cynical observation that, in free online services¹

you're not the customer. The ad service buyer is the customer. You're the commodity.

However, this isn't the whole truth.

In both commercial and academic systems, designers typically choose performance, price, and privacy trade-offs from a limited design space in which the price of usage is zero (the “freemium” model, which we discuss later, is an exception). We propose that by relaxing some built-in assumptions, we can find different trade-offs that give users more control over their privacy and require less trust in OSN operators.

Our goal isn't to guard user data from friends or governments but to reduce OSN providers' ability to disclose user data beyond users' wishes—without compromising functionality. This task is far from trivial. All changes come at a cost, and there's no guarantee of adoption, much less overturning today's market leaders. Nevertheless, we argue that the technical and economic aspects of the problem are surmountable.

The Problem of Privacy

Online service users have grown accustomed to not paying; but even “free” services have a cost. Users must give control over who sees what to a service provider that is beholden to other interests besides the users. This is the kernel of truth in the observation above. However, it's only a kernel: OSNs that disregard users' desires entirely can't succeed.

Who's the Boss?

It's an oft-repeated episode: a user shares information with friends via an OSN only to discover that this information has spread more widely than desired, with personal, professional, or even criminal consequences. We can prevent or mitigate such incidents by improving privacy usability (see “The Usability of Online Social Network Privacy” sidebar). This field of research can help users gain more control over their privacy, providing them with tools for better expressing their preferences and understanding the effects of their privacy choices.

Some OSNs have adopted privacy usability techniques, but basic privacy usability principles often conflict with their economic imperatives. For instance, the fundamental assumption that users should have full

The Usability of Online Social Network Privacy

Several developments in the privacy usability field offer users better control over the dissemination of their personal information. Many of these techniques could be applied in today's online social networks (OSNs) and in alternative architectures, such as Footlights (see the main text). Privacy usability is an active field; all the answers are not yet known. We list several approaches to privacy usability work, some of which have already been adopted by existing OSNs. Further adoption depends not only on the state of the technological art but also on OSNs' willingness to see it deployed.

Understanding Preferences

To improve the state of privacy in computing, we should understand users' privacy preferences. Janice Tsai and her colleagues showed that providing users feedback increases their comfort in sharing location data with friends,¹ and Jennifer King and her colleagues² and Maritza Johnson and her colleagues³ have explored the real-world understanding and use of Facebook's privacy settings.

Feedback in Practice

Heather Lipford and her colleagues explored the feedback principle in their Facebook-based "Audience View" work,⁴ and Andrew Besmer and his colleagues expanded on it with their social information about OSN applications.⁵ Na Wang and her colleagues have similarly attempted to provide OSN users with more information about what applications can do with their data,⁶ whereas Alessandra Mazzia and her colleagues' PViz attempted to align OSN privacy comprehension with users' mental models.⁷

User Expression

Another vein of privacy usability research focuses on improving systems' ability to learn what settings users want. Sameer Patil and Jennifer Lai provided a means for users in a workplace-focused social networking trial to specify privacy preferences at different granularities and found that groups were a favored level of abstraction.⁸ Luke Church and his colleagues allowed OSN users to express their privacy preferences via programming-like concepts, such as user-defined abstractions.⁹ Policy specification UIs were joined by machine learning techniques in Norman Sadeh and his colleagues' work on mobile privacy preferences in a location-based social network¹⁰ and Lujun Fang and Kristen LeFevre work with Facebook preferences.¹¹

References

1. J.Y. Tsai et al., "Who's Viewed You?: The Impact of Feedback in a Mobile Location-Sharing Application," *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI 09)*, ACM, 2009; <http://portal.acm.org/citation.cfm?id=1518701.1519005>.
2. J. King, A. Lampinen, and A. Smolen, "Privacy: Is There an App for That?" *Proc. 7th Symp. Usable Privacy and Security*, ACM, 2011; http://cups.cs.cmu.edu/soups/2011/proceedings/a12_King.pdf.
3. M. Johnson, S. Egelman, and S.M. Bellovin, "Facebook and Privacy: It's Complicated," *Proc. 8th Symp. Usable Privacy and Security*, ACM, 2012; <http://portal.acm.org/citation.cfm?id=2335356.2335369>.
4. H. Lipford, A. Besmer, and J. Watson, "Understanding Privacy Settings in Facebook with an Audience View," *Proc. 1st Conf. Usability, Psychology, and Security*, Usenix, 2008, pp. 1–8; www.usenix.org/events/upsec08/tech/full_papers/lipford/lipford.html.
5. A. Besmer et al., "Social Applications: Exploring a More Secure Framework," *Proc. 5th Symp. Usable Privacy and Security*, ACM, 2009; <http://dl.acm.org/citation.cfm?id=1572535>.
6. N. Wang, H. Xu, and J. Grossklags, "Third-Party Apps on Facebook: Privacy and the Illusion of Control," *Proc. 5th ACM Symp. Computer Human Interaction for Management of Information Technology*, ACM, 2011; <http://portal.acm.org/citation.cfm?id=2076444.2076448>.
7. A. Mazzia, K. LeFevre, and E. Adar, "The PViz Comprehension Tool for Social Network Privacy Settings," *Proc. 8th Symp. Usable Privacy and Security*, ACM, 2012; <http://portal.acm.org/citation.cfm?id=2335356.2335374>.
8. S. Patil and J. Lai, "Who Gets to Know What When: Configuring Privacy Permissions in an Awareness Application," *Proc. SIGCHI Conf. Human Factors in Computing Systems*, ACM, 2005; <http://portal.acm.org/citation.cfm?id=1054972.1054987>.
9. L. Church et al., "Privacy Stories: Confidence in Privacy Behaviors through End User Programming," poster, *Proc. 5th Symp. Usable Privacy and Security*, ACM, 2009; <http://cups.cs.cmu.edu/soups/2009/posters/p3-church.pdf>.
10. N. Sadeh et al., "Understanding and Capturing People's Privacy Policies in a Mobile Social Networking Application," *Personal and Ubiquitous Computing*, vol. 13, no. 6, 2009; <http://portal.acm.org/citation.cfm?id=1569346.1569366>.
11. L. Fang and K. LeFevre, "Privacy Wizards for Social Networking Sites," *Proc. 19th Int'l World Wide Web Conf.*, ACM, 2010; <http://dl.acm.org/citation.cfm?id=1772727>.

control over how their information is shared is at odds with OSNs' growth imperative: information sharing attracts new users, so some minimum level of sharing is usually required ("See, your friends are all here and having a good time!").

OSNs have some incentive to accurately capture

users' intentions for sharing: if users are "burned" by too many unintended disclosures and feel unsafe in the network, the OSN might have difficulty fulfilling its growth imperative. Nonetheless, we've observed OSNs changing users' default privacy settings to share information that users had previously designated private.

Social Applications

In many social applications, users can't express privacy preferences, and sometimes, the preferences they do express are ignored. For instance, when an author of this article joined Facebook in 2006, he elected to restrict access to his friends list. This list was private, first, because it was potentially embarrassing that some acquaintances weren't on it, and second, because private attributes (such as age, religion, and political affiliation) can be inferred from those of friends, as Wanhong Xu and his colleagues' re-

search shows.² However, despite this, Facebook later decided that friends lists would be shared freely with application developers, ironically announced while "launching new tools

to give you even greater control over the information you share."³ After a backlash, users can now hide some "public information" from other Facebook users, but not from applications. Other services, such as LinkedIn and Google+, also require some information to be publicly visible; users have incomplete control over their data.

OSNs are *two-sided markets*: more applications attract more users, and more users attract more application developers. Services need to attract developers, and generous access to user data provides a compelling case. OSNs differ in how much information they automatically share with applications; for instance, Facebook includes automatic preauthorized sharing with "instant personalization" partners (at least one of whom has demonstrably abused its trusted position⁴). When OSNs share user data with applications, they typically impose no technical restrictions on what applications can do with the data.

OSNs and application developers might have commercial relationships that contractually forbid developers from using private user data in certain ways. Violating contracts and user expectations could damage a firm's reputation, but this doesn't seem to decrease user counts or real value. In 2010, developers of some of the most popular—and most trusted—OSN applications were observed passing users' private data directly to "dozens of advertising and Internet tracking companies."⁵ The revelation of this behavior hasn't dissuaded millions of users from playing FarmVille, and the damage to owner Zynga's reputation was minimal. If contract and reputation weren't enough to prevent this behavior, why aren't technical methods of control imposed on third-party applications?

Application confinement, or *sandboxing*, requires

that software run on a trusted computing base under the control of the party enforcing the confinement policy. If an OSN confined third-party applications' behavior on behalf of users, application code would need to run on computers controlled by the OSN or the users themselves. Such an approach has the potential to provide much better enforcement of users' privacy goals but at a cost to both platform and application developers. Confinement systems that prevent undesirable effects without impinging on desirable function-

ality and performance are significant undertakings with significant costs. Imposing confinement might also increase the coordination costs between platform and application developers, possibly precluding devel-

opers from carrying out real-time application updates and forcing them into a more structured, app store-like model.

Running applications primarily with OSN-controlled resources would change incentives. If today's applications use inefficient algorithms, developers will bear any excess computational cost because they're responsible for running their own code. If OSNs execute applications on behalf of users, they will incur computing resources costs. If OSNs attempt to recover these costs from developers, this will require accounting and billing overhead that would increase costs (in both time and money) and decrease developers' flexibility, making the platform less attractive.

Choice in Advertising

In the case of targeted advertising, users aren't given the chance to make privacy choices. This might be fine if targeted ads leaked no information to the advertiser; however, Aleksandra Korolova found that they do, even if users only view them.⁶ Facebook has mitigated this issue, but ineffectually; it could take more effective measures, but these would reduce the value of its targeted advertising. Balachander Krishnamurthy and Craig E. Wills have previously observed OSNs such as LiveJournal and hi5 directly leaking age, gender, email address, and even postal codes to advertisers (whereas Facebook does not).⁷ The price of targeted advertising on the Web is also supported by enforcement mechanisms, such as frame-busting, that defend against "click fraud" with rootkit-like techniques that require complete trust by users' Web browsers.

All these privacy failures occur because of misalignments between user interests and OSN incentives.

When OSNs share user data with applications, they typically impose no technical restrictions on what applications can do with the data.

Users might not be the product, but when a service is paid for by others, they aren't the customers either.

Paying for Privacy

Incentive misalignment isn't unique to OSNs. Many businesses subsidize their services with revenue from advertising or by sharing customer data, for example, using shared customer loyalty schemes. In some cases, this data sharing comes cheap. In one detailed study, Nicola Jentzsch and her colleagues found that users were willing to provide detailed

personal information to a researcher-run online movie ticket vendor in exchange for a discount of €0.50.⁸

This builds on Jens Grossklags and Alessandro Acquisti's earlier

finding that "most subjects happily accepted to sell their personal information even for just 25 cents."⁹ If people are willing to trade their personal information for small subsidies, we might reasonably ask whether users are willing to pay anything for privacy-enhancing services.

The studies cited earlier are just a few examples of the apparently low value that people place on privacy. If users' privacy behaviors aren't economically rational, as Acquisti has claimed, then providing users with more information about what happens to their personal information and the potential harm that disclosure can cause might be insufficient to convince them to pay for privacy.¹⁰ However, Jentzsch and colleagues' work provides evidence for the alternative hypothesis: users *are* willing to pay for privacy. In their study and elsewhere, users have paid for privacy or to exclude themselves from targeted advertising.

Some services use a freemium model, letting users pay for a version of the service without advertising. This model is popular in mobile app stores; the blogging site LiveJournal; the audio streaming service Spotify; and the new, paid, centralized OSN App.net. The existence of these services suggests that many users are willing to pay to be rid of advertising—an important component of the privacy question. What's more, App.net describes part of its value proposition in terms of privacy and trust: "Many people have become so cynical about user-hostile, privacy-violating social services that they refuse to participate at all. We can understand why. Earning your trust is the most important thing we can do" (<https://join.app.net/#value-seven>). If services like App.net are successful, a market for Web services funded by means other than targeted advertising might be viable.

Still, even if the costs are as low as US\$1 per user

per year, as we describe later, some users will be unwilling to pay. These users could be subsidized by privacy-preserving advertising technologies (such as Adnostic¹¹) or by other users or organizations (for example, work use subsidizing personal use), but as we discussed, subsidies can cause incentive misalignment. We leave as an open question the size of the market that would be willing to pay for such a service and whether it could overcome existing service models.

Using a freemium business model doesn't necessarily provide privacy. In

LinkedIn, users who pay for enhanced services can see more information about other users than nonpaid "basic" users and can contact users outside

their immediate social net-

work. That is, those who pay LinkedIn are able to pull data about users (as applications do in conventional OSNs) and push data out to users (as advertisers do in conventional OSNs) beyond normal users' capabilities. Seen in this light, LinkedIn is actually not so different from conventional OSNs: two classes of economic actors interact with the system—one gets free service, and the other has privileged access to users.

Although the debate is by no means settled, the existence of paid, advertising-free services suggests that there might be a market for users willing to pay directly for their online usage. The assumption that future social technologies must be paid for by today's advertising limits the design space available for exploration.

Privacy versus Performance: Existing Approaches

There have been many attempts to provide OSN users with better privacy protection, but none have fully succeeded or garnered wide adoption. These approaches fall under two broad categories: scrambling user data in conventional, centralized OSNs and distributing user data over peer-to-peer networks. Both make an essential trade-off for free services, sacrificing privacy for performance or vice versa.

In the first type of solution, the OSN encrypts, permutes, or otherwise scrambles user data. Early examples of this type of system are Saikat Guha and colleagues' NOYB¹² and Matthew M. Lucas and Nikita Borisov's flyByNight.¹³ In this class of systems, users sign in to a conventional OSN, but their photos and profile attributes are encrypted or stored elsewhere. This prevents the OSN operator from reading any particular user's private information.

“Your friends can modify their client software to share your details more widely than asked. But if they do, you don't need new technology; you need new friends.”

However, this type of solution doesn't actually provide users with effective privacy. Widespread use of these systems would lead to one of two outcomes: either the service would ban them because they break the OSN's business model, or—more troubling—the service would tolerate them because they don't break the business model. That is, although these systems might hide my political affiliation from the OSN operator, they can't hide the social graph: who my friends are, who their friends are, and so forth. Again, Xu and colleagues' work,² as well as that of others, has shown that given knowledge of the social graph, adversaries can calculate likely political affiliation and other personal attributes hidden by the encrypt-within-the-OSN class of systems. Usually, you are who your friends are.

The second category of solutions eschews centralized OSNs entirely. Instead, user data is stored in peer-to-peer networks that might have some degree of federation and might use encryption. An example of such a system is Leucio Antonio Cuttillo and his colleagues' Safebook, which distributes user data over a peer-to-peer overlay network and stores private information on friends' computers.¹⁴ For users to find my data, they must find a path to one of my friends through various "rings" in the overlay. Safebook's authors claim that this "matryoshka" routing scheme provides privacy properties, but the actual threat model and security claim are unclear. Building a reliable P2P network often requires exposing some amount of social graph information to the network.¹⁵ But it's the social graph that we most want to protect!

In addition, this scheme imposes a performance penalty. In a paper on Safebook, the authors describe the network conditions under which the system has 90 percent availability; that is, there's a 90 percent probability that a user will be able to access information another user provided.¹⁴ If found, the path connecting the two users might pass through network nodes with poor uplink speeds. However, even if the path has excellent bitrate and latency characteristics, there's a 10 percent probability that there won't be a path at all. This system won't compete well with existing commercial infrastructure wherein cloud providers often refund customers if availability drops below 99.9 percent.

In both these classes of systems, privacy and performance are in conflict. Systems that must fit into the model of centralized OSNs to exploit their resources fall short of the privacy goals that might be reached if designers had a free hand. Systems that use the free storage provided by peer-to-peer networks are subject to the capricious churn of unreliable network nodes. The conflict between privacy and performance is caused by

the underlying assumption that if a system is not gratis, people won't use it.

This economic assumption limits designers of new systems. By lifting this constraint, designers can explore a much more capacious design space, allowing system design that provides different performance, price, and privacy trade-offs.

Here Be Dragons: Unexplored Regions of the Design Space

When we approach the problem of privacy in OSNs without artificial constraints, we can design new classes of systems that provide very different results from those traditionally assumed possible.

Centralized or Distributed?

Centralized, commercial infrastructure allows free OSNs to deliver content with high availability and low latency. We can achieve more user control by keeping user data on users' machines, but there are performance penalties associated with creating large, distributed systems using home computers.

However, if we relax the price constraint on our design space, new alternatives begin to appear. We can retain performance benefits of a centralized infrastructure if the provider is paid explicitly (rather than "in kind" with private information) through a subscription model. Alternatively, OSN providers might offer corporate customers a social networking appliance that's held inside corporate firewalls, preventing data from leaking outside organizations.

If infrastructure providers are paid explicitly, users could store encrypted data on centralized storage and content delivery services. Curious OSN providers could still perform traffic analysis to obtain an unlabeled social graph; Arvind Narayanan and Vitaly Shmatikov (as well as others) have shown that such graphs can be reliably deanonymized given sufficient data.¹⁶ Still, a world in which paid providers conduct traffic analysis to infer social graphs is very different from today's world in which unpaid providers leverage user data to earn revenue from sources other than the users themselves.

In this semicentralized approach, infrastructure providers must get paid, but this cost need not be prohibitive (\$1/user/year).¹⁸ Users might be willing to directly pay such a low cost—a single premium text message could buy years of reasonable usage. Mobile networks, phone manufacturers, or OS vendors might choose to bundle a subscription to their products (for example, Amazon's WhisperNet and Apple's iCloud). The infrastructure costs of realistic systems can be quite low, but the key is that providers' incentives are aligned with those of users.

Another trade-off point might be found in a federated

architecture: users could store plaintext data at one of a number of providers, each of which have access to only part of the larger social graph, as in the Diaspora project (<http://diasporaproject.org>). If providers are paid, commercial infrastructure could be employed for high reliability without conflicting incentives. The challenge in such a model would be providing useful cross-domain functionality without compromising user data confidentiality. The risk of deanonymization by a determined adversary would still be present.

Safe Applications

Today's OSNs aren't merely data repositories; they're platforms for social applications. OSN providers support ecosystems of third-party application developers who provide users with new functionality not anticipated or designed by the original OSN designers. Such functionality should also be made available in any privacy-preserving OSN; the application platform shouldn't cause user data to flow beyond users' control without their permission.

One model for confined social applications could be to run them on the OSN's computers. However, as we described, this might lead to incentive misalignment: the OSN operator would bear the cost of running applications, but the developers determine the applications' efficiency. If the OSN operator exposed the true cost to run applications, competitive pressure might be brought to bear. A variation on this model could be the social appliance, described earlier, which coordinates the running of third-party applications on computers owned by the organization paying for the appliance.

A semicentralized approach to social storage, in which confidentiality is enforced locally on users' computers through cryptography, lends itself to application confinement on users' individual computers. Applications that run on a user's computer, operating on local data, can be confined using OS or language-level techniques, preventing them from contacting networked services or accessing unauthorized data. Once installed, applications could be given access to a security API that defaults to indirect manipulation of user data and is governed by explicit expressions of user intent, as Ka-Ping Yee describes in "Aligning Security and Usability."¹⁷ For instance, a photo-sharing application might be granted the authority to edit a copy of an image that the user explicitly drags and drops, but when sharing that image with a friend, the application doesn't need to know the name of either party. Instead, indirection can support the object-reference equivalent of "this user" and "that user."

The application platform's security kernel can be quite small. It might handle several standardized data

formats (specified in an RFC-like manner) and present a standardized security API, but an open source kernel can be written and maintained at little cost, like other open source projects. As with other open source projects, it can be scrutinized by many eyes, so users don't need to trust this kernel in the same sense as a centralized OSN.

Whatever shape the OSN application platform takes, third-party social applications will provide users interesting new functionality. These applications could be distributed via an app store that provides opportunities for application behavior to be vetted and for application authors to collect remuneration, supporting further development.

Use Cases

Users of more-private OSNs must be able to take actions provided by today's OSNs, such as finding each other and creating *joint content*.

Finding friends. We're often asked, "If an OSN doesn't have a centralized operator, how will I find my long-lost friends from school?" In a private OSN, finding other users can either be done with external, public data or with an explicitly social protocol.

Because no centrally enforced policy requires users and accounts to have a 1:1 mapping, we expect that some users will present multiple identities to a privacy-preserving OSN, which might include public personas. Finding the public persona of anyone with a personal, corporate, or academic webpage will be a matter of using a standard search engine.

Often, old friends are found on social networks as friends of friends, and most OSNs provide a Friends of Friends (or Extended Circles) setting that lets people connect online. Without centralized social graph mining, this use case is still simple to fulfill: users need only give their friends a "please tell your friends about me" token that can be used to generate friend suggestions like those that appear in today's OSNs. This is a social approach to a social problem.

Joint content. Another use of today's OSNs is the creation of joint content: digital artifacts with multiple stakeholders. Suppose that Alice uploads a photo, Bob tags it to say that Charlie appears in it, and Dave comments on Bob's tag. Who owns the content? Technology alone can't solve this sociotechnical problem, but it can help those involved to clearly express what they meant to express.

In this scenario, each user can upload his or her contributions separately, linking explicitly to a clear context and providing opportunities for other users to link to them. Alice uploads the photo and then updates her

album to link to the photo. Bob can then upload the tag of Charlie, but it would only be associated with the album if Alice updates her album to also include the tag. A useful convention might be that if a user hasn't signed a piece of content it shouldn't be regarded as really being him or her; so, Bob will also need Charlie (or Charlie's software, acting on his behalf) to attest to the tag. It's technically possible for your friends to modify their client software to share details more widely than asked. But if they do, you don't need new technology; you need new friends.

Finally, when Dave uploads his comment, he will link to the context that he commented on, preventing Alice from changing the apparent subject of Dave's "that's cool!" comment to some embarrassing image. Alice is always free to modify her own photo albums, but if she changes the photo, Dave's comment link will no longer apply.

This approach to joint content can also be extended to support groups of users sharing and discussing content.

Limitations

Any approach to online social networking that puts control back in users' hands will face some limitations. Today's "big data" analysis of social graph data is only possible because private user data is centralized at the provider. If providers can't inspect user data, they can't perform analyses, such as "What's trending among friends of friends of friends?" Some of this computation could be performed locally—on whatever raw data a user can access—but the cost to do so might be significant. For instance, many signals that can be used to promote content from "accessible" to "visible right now" are still available: who the content is from, how close users are in the network, how frequently they communicate, and so forth. Users could choose for their client software to share interests and activities with friends or even more widely, allowing various degrees of trend-spotting. Exactly how much benefit is derived for a certain expenditure of computational time and energy is an empirical question, an issue for future work.

A similar limitation exists for audit functionality. Whether it's used to detect copyright infringement or cyberbullying, centralized auditing of user content requires that all user content be visible to a trusted party, such as the OSN operator. The price of the security that these auditing mechanisms provide is confidentiality; no system that provides truly user-driven confidentiality can be open to systemic audits of content. Instead, enforcing copyright, detecting cyberbullying, or implementing other policies must be done in a distributed way.

Footlights

Having considered this unexplored region of the design space, we present one solution drawn from it. We have designed and implemented an architecture and open source prototype of a semicentralized OSN and application platform called Footlights.¹⁸

Footlights' design constraint is trust. Users shouldn't need to trust any third party with their private data. They might rely on third parties to perform a task or even verify that it's been performed correctly, but they need not expose themselves to risk of harm from said third parties. That is, if a user's private information is disclosed to friends, strangers, or advertisers, it should be because of a user's choice and not because of an undesired access control decision that an OSN provider was trusted to make on the user's behalf.

The Footlights approach marries local enforcement of users' privacy goals with centrally provided storage and content delivery networks. Applications run on users' computers, subject to sandboxing, and are provided with a security API for interacting with user data, other users, and other applications. Information stored on centralized systems is encrypted locally; systems rely on, but do not trust, the infrastructure.

A key Footlights feature is that it's compatible with existing optimizations—for both privacy and performance—at the network layer, and different users who choose different trade-offs can coexist in the same overall system.

Some traffic analysis can be mitigated with expensive route-obfuscation schemes, such as querying batch-oriented anonymizing proxies or fetching each block over a different Tor connection. These measures can't provide perfect anonymity: you can't act unilaterally in a social network, and even if all of your friends use these expensive measures, a motivated, well-equipped adversary might still observe the relative timing of block uploads and downloads. This timing information implies an unlabeled social graph that can be deanonymized given sufficient data,¹⁶ but the work required to do so is far greater than if users at well-known IP addresses directly upload and download blocks.

For some users, performance is king; they will prefer to access infrastructure via the local caches of content delivery networks. Data confidentiality is maintained, but traffic analysis of these users is almost trivial. Still, this choice belongs to users, not the OSN provider, and different users working together can make different choices.

The Footlights kernel runs as a local application on the user's computer, but it supports a familiar Web-based UI and can be started using existing techniques such as Java Web Start.

Untrusted Infrastructure

As a semicentralized approach, Footlights uses commodity cloud storage providers to hold user data. This data is broken into fixed-sized blocks and then encrypted so that infrastructure providers can neither read the content nor determine which blocks are part of the same files: all they see is a sea of identically sized (4-Kbyte) ciphertext blocks. An arbitrary number of users can share one global block store without access control, because the store is content addressed: the block's name is derived from its ciphertext, and it keeps this name no matter who uploads it. This means that, when uploading ciphertext blocks to the cloud, the question isn't, "Is this user authorized to upload content with this name?" but rather, "Has this block of storage been paid for?"

The cost of maintaining this storage at scale is surprisingly low. Documents from Facebook's IPO report that when it had 845 million active users, it stored 100 petabytes of photos and videos—an average of 115 mebibytes of photo and video content per user. Assuming that Footlights users will store similar quantities of photo and video content, the cost of storing and transmitting users' data with commercial infrastructure is less than \$1 per user per year.¹⁸ This price doesn't include the cost of developing the storage framework and application platform—currently available under an open source license—but it does pay for all the centralized infrastructure that Footlights requires to compete with the performance of today's OSNs.

A content-addressed store allows for large quantities of data to be shared and consistently cached. Because names are derived from content, any name refers to an immutable snapshot of data. Of course, an immutable content-addressed store isn't enough to support practical applications: real applications require mutable names. For instance, a photo-sharing application needs to know about more than Peter's previous photo collection; it needs to know about current photo albums, their content, and therefore their content-derived names. Footlights supports this behavior by allowing mutable names to be resolved via standard Web mechanisms such as JSON-over-HTTP. That is, a user can publish a URL such as `www.cl.cam.ac.uk/~jra40/footlights` that contains a reference to a content-addressed block and a digital signature. Other users' client software can download this signed reference, check it against the last version, and then download the new version if necessary.

Local Access Control

Footlights' access control is performed locally on users' computers using cryptography, driven by users'

expressions of intent. For instance, when they "share" with friends, Footlights reveals encryption keys to the client software of the chosen users only.

The local Footlights client software interprets blocks that have been shared with it (or that it has created) as a file system, subsets of which can be exposed to applications through a security API. Figure 1 shows the layers of this file system and application stack.

Local, Distributed Applications

Footlights provides a platform for social applications with an API that lets applications work with user data but not leak it beyond the user's consent. Sharing with other users appears to work like it does in today's social networks but is backed by cryptographic mechanisms. Applications can bundle files in a directory to be shared with other users or applications, but the sharing itself is done by the Footlights platform, and only with explicit expression of user intent. The platform doesn't ask, "Application X wants to share data with user Y; is this permissible?" Instead it asks, "With whom do you want to share this data?"

As another example, instead of copying users' private information to display it back to the user, applications can use indirection and placeholders, that is, "put the user's name here." This indirection is based on Adrienne Felt and David Evans' privacy by proxy scheme.¹⁹ Applications can access data such as photos directly if the user explicitly expresses intent—for example, by choosing a photo in a dialog box—but even this level of sharing is more controlled than in today's OSNs. A photo-sharing application might be able to see users' photos and apply filters to them, but if it's unaware of the users' names and can't communicate directly with the outside world, their private information remains very much private. The system is designed to facilitate widespread sharing of arbitrary quantities of data, but nothing is shared with other users or applications unless the user directs it. In this way, users' local sharing decisions are projected into globally shared infrastructure without revealing who is sharing with whom.

Using untrusted infrastructure together with local access control, Footlights provides features similar to traditional OSNs. However, users control how the system shares their information, subject to the practical realities of traffic analysis and gossip.

Footlights is one example of a system that can be built in the previously unexplored portion of the available design space. It proves that alternative OSNs can provide privacy and performance while being both technically and economically viable. However, the fact

that a system with better privacy properties can be built doesn't, in itself, make it a practically available alternative: Metcalfe's law of network effects applies; it's hard to get anyone to join an empty social network. Nonetheless, we've demonstrated that this alternative is possible. The way in which today's incumbent OSNs operate is just one way of doing business. Privacy isn't inherently incompatible with social networking. ■

References

1. Liorean, blog, 2004; <http://codingforums.com/showpost.php?p=206569>.
2. W. Xu, X. Zhou, and L. Li, "Inferring Privacy Information via Social Relations," *Proc. IEEE 24th Int'l Conf. Data Engineering Workshop*, IEEE CS, 2008, pp. 525–530; <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4498373>.
3. R. Sanghvi, "New Tools to Control Your Experience," Facebook blog, 9 Dec. 2009; <https://www.facebook.com/blog/blog.php?post=196629387130>.
4. P. Ganapati, "Scribd Facebook Instant Personalization Is a Privacy Nightmare," *Wired*, 30 Sept. 2010; www.wired.com/epicenter/2010/09/scribd-facebook-instant-personalization.
5. E. Steel and G.A. Fowler, "Facebook in Privacy Breach," *Wall Street J.*, 17 Oct. 2010; <http://online.wsj.com/article/SB10001424052702304772804575558484075236968.html>.
6. A. Korolova, "Privacy Violations Using Microtargeted Ads: A Case Study," *Proc. 10th IEEE Int'l Conf. Data Mining Workshops (ICDMW 10)*, IEEE CS, 2010, pp. 474–482; <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5693335>.
7. B. Krishnamurthy and C.E. Wills, "On the Leakage of Personally Identifiable Information via Online Social Networks," *Proc. 2nd ACM Workshop Online Social Networks (WOSN 09)*, ACM, 2009; <http://dl.acm.org/citation.cfm?id=1592665.1592668>.
8. N. Jentzsch, S. Preibusch, and A. Harasser, *Study on Monetising Privacy*, tech. report, Feb. 2012; www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/monetising-privacy.
9. J. Grossklags and A. Acquisti, "When 25 Cents Is Too Much: An Experiment on Willingness-to-Sell and Willingness-to-Protect Personal Information," *Proc. 6th Workshop Economics of Information Security (WEIS 07)*, 2007; <http://weis2007.econinfocsec.org/papers/66.pdf>.
10. A. Acquisti, "Privacy in Electronic Commerce and the Economics of Immediate Gratification," *Proc. 5th ACM Conf. Electronic Commerce (EC 04)*, ACM, 2004; <http://portal.acm.org/citation.cfm?id=988772.988777>.
11. V. Toubiana et al., "Adnostic: Privacy Preserving Targeted Advertising," *Proc. Network and Distributed System Security Symposium (NDSS 10)*, Internet Soc., 2010; www.isoc.org/isoc/conferences/ndss/10/pdf/05.pdf.
12. S. Guha, K. Tang, and P. Francis, "NOYB: Privacy in Online Social Networks," *Proc. 1st Workshop Online Social Networks (WOSN 08)*, ACM, 2008; <http://portal.acm.org/citation.cfm?id=1397735.1397747>.
13. M.M. Lucas and N. Borisov, "FlyByNight: Mitigating the Privacy Risks of Social Networking," *Proc. 7th ACM Workshop Privacy in the Electronic Society (WPES 08)*, ACM, 2008; <http://portal.acm.org/citation.cfm?id=1456403.1456405>.
14. L.A. Cutillo, R. Molva, and T. Strufe, "Safebook: Feasibility of Transitive Cooperation for Privacy on a Decentralized Social Network," *Proc. 3rd Int'l IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC 09)*, IEEE, 2009, pp. 1–6; <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5282446>.
15. B. Viswanath et al., "An Analysis of Social Network-Based Sybil Defenses," *Proc. ACM SIGCOMM 2010 Conf. Data Communication (SIGCOMM 10)*, ACM, 2010; <http://dl.acm.org/citation.cfm?id=1851182.1851226>.
16. A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," *Proc. 29th IEEE Symp. Security and Privacy (SP 08)*, IEEE CS, 2008, pp. 111–125; <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4531148>.

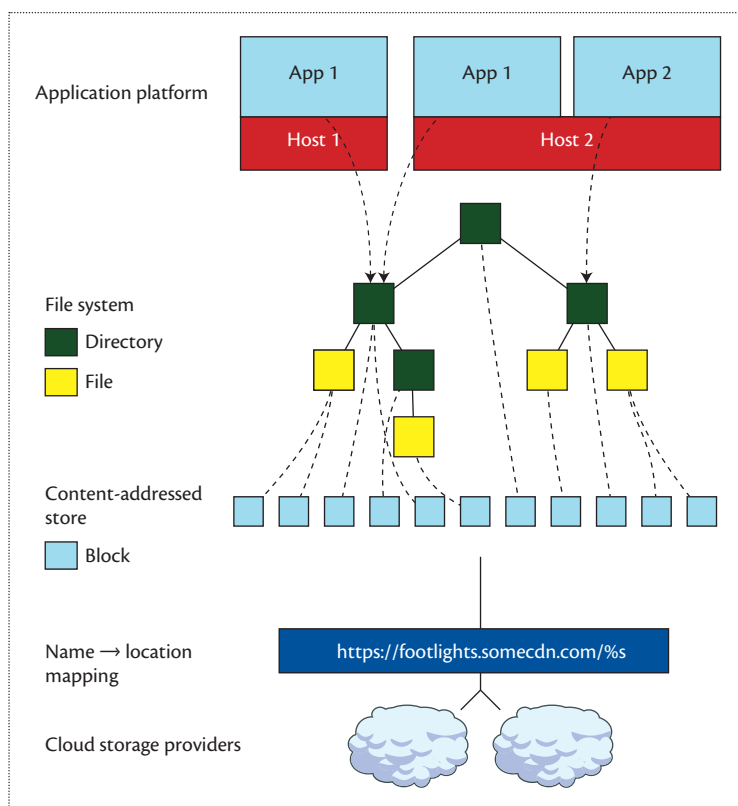


Figure 1. File system layers visible to applications. Storage providers (at bottom) serve blocks of content but can't see their structure or relationships.

17. K.-P. Yee, "Aligning Security and Usability," *IEEE Security & Privacy*, vol. 2, no. 5, 2004, pp. 48–55; <http://doi.ieeecomputersociety.org/10.1109/MSP.2004.64>.
18. J. Anderson, "Privacy Engineering for Social Networks," PhD dissertation, Computer Laboratory, Univ. Cambridge, 2012; www.dspace.cam.ac.uk/handle/1810/244239.
19. A. Felt and D. Evans, "Privacy Protection for Social Networking Platforms," *Web 2.0 Security and Privacy (W2SP 08)*, IEEE, 2008; <http://w2spconf.com/2008/papers/s3p1.pdf>.

Jonathan Anderson is a postdoctoral research associate at the University of Cambridge. His research interests are in the intersection among operating systems, applications, security, and privacy. Anderson received a PhD in computer security from the University of

Cambridge. He's a member of IEEE. Contact him at jonathan.anderson@cl.cam.ac.uk.

Frank Stajano is a faculty member in the University of Cambridge Computer Laboratory security group. His personal grand challenge is making the digital society fair for nongeeks—this motivates his research interests in privacy; the psychology of security; and specifically, liberating computer users from passwords. Stajano received a PhD in computer security from the University of Cambridge. He's a Toshiba Fellow and the author of *Security for Ubiquitous Computing* (Wiley 2002). Contact him at frank.stajano@cl.cam.ac.uk.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

Call for Papers

Moving-Target Defense

for *IEEE Security & Privacy* magazine's March/April 2014 issue

Abstracts due to the guest editors:

1 June 2013

Articles due to ScholarOne: 1 July 2013

Hitting a moving target is usually more difficult than hitting a stationary one. In World War II, naval ships zigzagged through the water to make it harder for submarines to torpedo them, and Hedy Lamarr and George Antheil's invention of frequency-hopping eventually made radio communications harder to jam. But some defensive techniques—like zigzagging—are soon negated by effective countermeasures. So how can we embrace a moving-target defense that has promise for long-term effectiveness?

Moving-target defense in cyberspace has been an announced priority for research programs for several years, and increasing numbers of techniques have been proposed and some (such as ASLR) have been widely deployed. This special issue of *IEEE Security & Privacy* magazine seeks papers that characterize the state of the art and future directions in moving-target defense.

We welcome case studies, experience reports, practices, research results, and standards reports. Our readers are eager

to hear about industry experiences, especially resulting from empirical studies that help us learn how past successes and failures should inform the next generation.

Submission Guidelines

Submissions will be subject to the IEEE Computer Society's peer-review process. Submission of an abstract, one page or less, is strongly encouraged. Articles should be at most 6,000 words, with a maximum of 15 references, and should be understandable to a broad audience of people interested in security and privacy. The writing style should be down to earth, practical, and original. Authors should not assume that the audience will have specialized experience in a particular subfield. All accepted articles will be edited according to the IEEE Computer Society style guide. Submit your papers to Scholar One at <https://mc.manuscriptcentral.com/cs-ieee>.

Questions?

Contact guest editors Luanne Goldrich (Johns Hopkins University Applied Physics Laboratory, Luanne.Goldrich@jhuapl.edu) and Carl Landwehr (George Washington University, Carl.Landwehr@gmail.com).

www.computer.org/security/cfp

