ABSTRACT

Online social networks — such as Facebook and Twitter — have become an essential tool for communication. Their ease-of-use and the associated network effect have made them ubiquitous. Unfortunately, they often do little to protect user privacy. Instead, they frequently sell user information to advertisers, and provide it to governments when requested. They comply with government requests for censorship, and in authoritarian states, they may be blocked entirely. This is clearly unacceptable, especially since the communication these networks facilitate often inspires needed social change.

We propose Identikey, an encrypted and distributed social network. Instead of the conventional model — where user data is stored by a central entity (e.g. Facebook) — Identikey stores user data across the machines of network participants. Because the data is distributed, it cannot be easily censored or modified by the central entity. Since it is encrypted, only users intended to receive content are able to access it. We believe this system is much more censorship resistant than other social networks, while preserving user privacy.

1. INTRODUCTION

Online social networks have historically relied on a centralized client-server model. When a user connects to one of these networks, he or she opens a web browser or mobile application, and that application connects to the social network, typically using the Hypertext Transfer Protocol (HTTP).

There are several computers (routers) between the user and the network’s server that have access to data and metadata communicated between the user and the network. There are also additional links between the network’s web servers, its file servers, its database servers, etc.

Unfortunately, this model is inadequate to protect user privacy. An adversary, whether a “hacker”, cyber-criminal, abusive government, or other abusive regime, can attack this network at several points:

1. Between the user and the network’s web servers — the adversary can either eavesdrop on the connection between the user and the network, or launch an active man-in-the-middle (MITM) attack. In both cases, the adversary gains access to the user’s communications and associated metadata. This is somewhat mitigated by HTTPS, which encrypts communications between the user and the network. However, HTTPS does not hide all associated metadata, and a sophisticated adversary could obtain a illegitimately-signed certificate to initiate a MITM attack.

2. Between the network’s web servers and it’s file/database servers — the adversary can eavesdrop on the internal connections between the network’s servers. Since data on trusted internal networks is often not encrypted, the adversary could obtain significant amounts of user data (in 2013, it was revealed that the United States National Security Agency (NSA) conducted such an attack on Google, and other large companies, obtaining user data sent between data centers[4]).

3. On the network’s servers — an adversary could install malware on the network’s servers, either by exploiting a software vulnerability, or by obtaining access to the servers (malicious employee, stolen administrative credentials, etc). This malware can alter user data, or send it to an unauthorized third party.

4. Through administrative process or other coercive action — a government could force the network to divulge user data, threatening the network with serious (and potentially illegitimate) penalties if they do not comply. Similarly, a non-government entity can extort the network or its operators, forcing the illegitimate disclosure of data.

Additionally, the network may willingly inspect, audit, and disclose user data to third parties, without the user’s explicit knowledge. Although networks often claim this data is
anonymized or only viewed by algorithms, de-anonymization is often trivial. The information might also be sold to other companies without the user’s explicit consent.

Finally, an adversary interested in censoring communication might block access to the network entirely. Since the network is centralized, it’s very simple for the adversary to block all connections to a certain range of network addresses belonging to the network. This technique is often seen in oppressive regimes, especially during periods of expected social change or instability. It has most notably been employed in several Middle Eastern and East Asian countries, including China, Egypt, Syria and Iran.

To combat all of these issues, we created Identikey, a fully-distributed social network. In Identikey, all shared content is cryptographically signed, and encrypted to its intended recipients prior to being transmitted. This prevents an eavesdropping adversary, and any user or organization with whom the content was not shared from obtaining the content. Since all of this data is signed, no user can masquerade as another. Furthermore, since the network is fully distributed, an adversary cannot easily censor the network by blocking a pre-defined range of network addresses, or domain names.

2. RELATED WORK

There are currently several decentralized (federated) peer-to-peer online social networks available, including GNU Social, Diaspora, and Friendica, but there are no (published) fully distributed ones. There are also several examples of distributed networks that serve other purposes, such as GNUNet, FreeNet, Gnutella, and BitMessage.

Diaspora is a social network that a user can run on his or her own web server. The user can also choose to join an instance run by a friend, or other trusted third party. As a network that contains several hundreds of thousands of users Diaspora shows the power of maintaining a distributed network at a scalable level. However, as a federated service there is no censorship resistance for each individual instance, nor is there a privacy guarantee for users communicating across instances. The instance operator for Diaspora and similar federated online social networks can monitor the interactions between users, remove content, forge content, and restrict access to the network. A distributed network eliminates the possibility of having any centralized entity.

BitMessage is a decentralized, encrypted peer to peer system that uses a trust-less communication protocol. It avoids problems such as network flooding by using a Proof of Work system model which only allows messages to be sent after the average computer has expended, on average, four minutes of work to solve a computational problem. This is not suitable for a large social network, as each piece of content shared would require significant processing time. Usability and user retention is also decreased as user interaction is not in a familiar social network format.

Abbas et al. propose a gossip based distributed social networking system which relies on establishing friendship links among peers based on dynamic conditions of peer availability [2]. They argue that the rate at which peers join and leave the network, or the peer churn rate, is very high. Therefore, the construction of these friendship links is the foundation for cooperatively building the network to a larger scale. This implementation is based on Tribler, which is a peer to peer client that encrypts a key that the peer receiving the data has to decrypt for authentication [3]. However, it does not provide a means for signature verification, in which the identity of the sender of the data is verified based on a signature generated when there is a mutual relationship between the two users. Another limitation of the work is it does not create real, identifiable entities within the broader context of a social network. Though it is fully distributed, it does not provide the typical usability of a conventional social network.

Nilizadeh et al. developed Cachet, a decentralized architecture for social networking with caching. It uses the features of the Kademlia Distributed Hash Table to connect confidential identities across the network and provide information about the availability of data to be shared. [6] It involves a functional model that constructs a news feed which contains container objects, which can be a status, web URL, photo, or a collection of other container objects. Access to this content is maintained by trusted associations with other peers in the network. The trust is established by a layer of cryptography, which is based on a traditional public key and attribute based encryption scheme.

3. SYSTEM MODEL

3.1 Identities

The most important component of Identikey is the secure representation of identities. When identities are represented securely, one can be sure that he or she is sharing content with only those whom he or she has intended. We achieve this using asymmetric cryptography, modeled off of the OpenPGP standard. Each user “registers” with the network by generating a keypair, which includes a 4096-bit public and private key for signing operations, and a 4096-bit key for encryption operations. The two are separate to mitigate an erosion of safety guarantees creating by using one key for both purposes. 4096-bit RSA was chosen, as it’s presumed to offer the highest level of security among widely supported OpenPGP key types. In the future, a smaller (with equivalent security) Ed25519 key might be used, but Ed25519 is not yet widely supported by OpenPGP implementations.

The user’s key, which we will label their “identity” is then shared with other users on the network. We’ve created a modular backend for key sharing, so identities can be distributed over the network’s DHT (see below), SKS (PGP) keyservers, or any other suitable system.
Once a user obtains another user’s identity, he or she can begin sharing encrypted content with that user. He or she can also verify the authenticity of content shared by that user, since the content will be cryptographically signed using that identity.

A user can confirm a user’s identity through a “following” process. When a user follows another user, he or she verifies that this identity belongs to the user that it claims to belong to. In the background, a certification signature is created to cryptographically represent the relationship. After enough users have verified each other, a web-of-trust is established through the network of mutual friends. When a user searches for an identity that he or she hasn’t yet followed, a list of mutual friends will be displayed. If the list is extensive, the user can assume with some certainty that the identity they are viewing is legitimate.

3.2 Distributed Hash Table

Identikey’s Distributed Hash Table (DHT), uses a modified version of the Kademlia [5] DHT protocol, called signed-kademlia. In addition to the features and guarantees provided by Kademlia, signed-kademlia creates “namespaces”, which are based on a user’s identity fingerprint. Only a user possessing the identity’s keys may update DHT entries in the corresponding namespace, preventing an adversary from censoring or altering shared data. All of these entries are also signed, to ensure their authenticity. If a node receives an update with a bad cryptographic signature, or receives an update signed using a key other than the one specified by the namespace, it will reject the entry entirely, and prevent it from being distributed. If these illegitimate update requests continue, the peer will be blocked for a period of time to ensure network stability. If an adversary attempts to overwhelm the DHT with arbitrary data, they will be unsuccessful, as the DHT limits the size of updates.

Identikey initially integrates itself with the network by connecting to one of three bootstrap servers. These bootstrap servers are simply DHT participants with known static addresses. Although this permits some attempts at censorship, it’s not a single point of failure (see below). Centralized bootstrap servers are an unfortunate yet necessary trade-off when using a DHT-based system, and similar distributed systems are all subject to the same constraints (e.g. BitTorrent, Tor, I2P, etc).

It’s very difficult to prevent users from connecting to this DHT, without deep packet inspection. There is no complete list of peers, so an adversary cannot simply block the network addresses of all participants. The ports used by each user also also inconsistent, so a port-based blocking scheme would not work. Blocking the DHT bootstrap servers is possible, but once the user has connected for the first time, Identikey saves a list of known peers which it connects to during future attempts. The user can also manually specify a peer to join the network if necessary.

3.3 Sharing Content

All content shared on Identikey is shared using a two part process.

3.3.1 Content Descriptor

When content is first set to be shared, an entry is published in the DHT describing the content. The entry includes the type of the content, a hash of the content, the hashes of any content it references, and identities of users the content has been shared with. The content itself is not stored in the DHT, so the network is not overwhelmed by large files, such as photos and videos. The content descriptor is encrypted to the recipients of the content, but the recipients are not visible until the descriptor is decrypted. This prevents anyone other than the recipients from obtaining the list of recipients.

3.3.2 Content Distribution

After the content descriptor has been published to the DHT, followers who regularly poll the DHT will find the descriptor. If they decrypt it successfully, the content has been shared with them. To download it, they will request it from any of the users who are listed as recipients. If any of these users are online, the content will be transferred between the two parties over HTTP, signed and encrypted to all potential recipients. Once the download has completed, the content will be decrypted and transferred to the frontend once requested. If none of the original recipients are online, the content will be downloaded as soon as it’s available. This structure was chosen to maintain scalability of the network; participants only store data for content which they have an interest in.

3.4 Structure

Identikey is intentionally divided into two distinct systems, the frontend, which is responsible for all user interaction, and the backend, which is responsible for participating in the distributed hash table, all cryptographic operations, and sharing content with peers. The two components interact using an application program interface (API). This distinction was created to permit alternative frontends to be easily incorporated into Identikey, and permits developers to design new systems (potentially for other uses) based on Identikey’s distributed model. Accordingly, the backend is capable of being run by itself.

3.5 Frontend Usability

Identikey includes a default frontend, which is very similar to existing social networks. It allows users to search for friends, add friends, and share content with other users on the network in a familiar way. Since usability is one of the biggest hurdles to the adoption of secure systems, usability
was emphasized, and the resulting interface is very easy to use.

4. SYSTEM IMPLEMENTATION

As mentioned earlier, Identikey is broken down into two halves: the frontend and backend.

4.1 Frontend

The frontend has been designed using the Python programming language. It uses Flask, a Python micro-framework, to render pages to the user and respond to user input. All operations which require network interaction are offloaded to the API, by sending JSON over HTTP POST requests. Returned results are stored by the frontend, using the SQLite relational database, and the SQLAlchemy object-relational mapper (ORM).

4.2 Backend

Like the frontend, the backend has been designed using the Python programming language. Since asynchronous operations are essential to the performance of the network, the backend makes heavy use of Twisted, an event-driven networking framework. We use the Twisted application framework to run the daemon, which is responsible for maintaining the (signed Kadmelia) distributed hash table and accepting and executing commands from the API. The DHT peers interact with each other using RPC over UDP. The Twisted application runs an HTTP server for the API, which accepts commands sent as JSON via HTTP POST requests. Responses are returned as JSON over HTTP. All cryptographic operations are performed using gpglib, a Python library forked from python-gnupg. gpglib uses GnuPG — an implementation of the OpenPGP standard — to sign data, encrypt data, and certify other user’s keys (identities).

4.2.1 Bootstrap Servers

The bootstrap servers are used to allow a new user to find peers upon his or her first connection to the network. The network addresses of these servers are hard-coded into Identikey, but the user may specify an address manually in the case of censorship. The bootstrap servers run a slightly modified daemon, which participates in the DHT but does not accept the usual API commands. A Docker image for this daemon has been created, so any user can easily run a backup bootstrap server without too much configuration. In the future, these additional bootstrap servers (run by volunteers) could be hard-coded into Identikey to increase censorship resistance and reliability.

5. FUTURE WORK

When Identikey was engineered, there were several trade-offs made between performance, usability, and security. Accordingly, Identikey is not necessarily protected from every type of attack that an adversary can attempt against the network. Although the network is stable, an adversary with enough computing power could potentially affect reliability by joining the network for a bit, and then leaving it en masse. Future versions would include workarounds to ensure that data is maintained in this situation, by having peers re-send their data if such a drop is detected. It is also possible to determine that a person is using Identikey, even if it is not possible to determine how they are using it. Future version will allow proxying (with Tor, for example), to prevent this.

There is also potential for the Identikey backend to be used for non-typical social networks which require guarantees of security and reliability. For example, extensions could be developed to share medical records between facilities or doctors, or for sharing arrest records between law enforcement agencies.

6. ETHICAL CONSIDERATIONS

Identikey provides users with a privacy-maintaining, censorship-resistant online social network. Identikey’s design provides privacy for those who wish to participate in social networks without the fear of monitoring. This design is consistent with the assumption of a universal human right to privacy. The Universal Declaration of Human Rights, adopted by the United Nations General Assembly in 1948, states in Article 12 that "No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks.” [1] While there is no legal right to privacy explicitly defined in the United States Constitution, the Supreme Court, through its decisions in cases such as Griswold v. Connecticut, have ruled that the Constitution implicitly grants a right to privacy against government intrusion.

However, with the rise of non-discriminating nationwide online surveillance (conducted by government entities such as the National Security Agency), and the proposal and enactment of privacy-eroding legislation, the extent of this right to privacy against the government has come into question. The design decisions behind Identikey protect it from most forms of surveillance, and since no one party has access to all of the unencrypted network data, it would be impossible for government to order any entity to release user information.

This creates a potential for abuse: individuals with ill intent could use Identikey for illicit, immoral, or other unsavory purposes. Similar to services like Tor, some users of Identikey could take advantage of Identikey’s privacy guarantees to conduct illegal activity. However, this is an inescapable product of this type of system, and the benefit provided to the average user outweighs the impact of malicious users. If criminal activity does occur and police intervention is necessary, police can use conventional policing methods, and install malware on the targeted user’s computer to obtain his or her data. However, they will not be able to monitor user content without resorting to these methods.

Finally, one must consider their environment before using Identikey. Although Identikey does protect the contents of communications, connecting to Identikey in some jurisdictions may be criminal, or may be considered evidence of criminal wrongdoing. All users should ensure they understand the risks of this association.

7. CONCLUSION

By leveraging the power of distributed hash tables and public key cryptography, the Identikey team was able to successfully design and implement a censorship-resistant, privacy-maintaining social network. The implemented fron-
tend maintains ease-of-use, and the network functions as expected. To the user, Identikey will seem familiar, to adversaries, however, it will seem anything but.

8. REFERENCES


