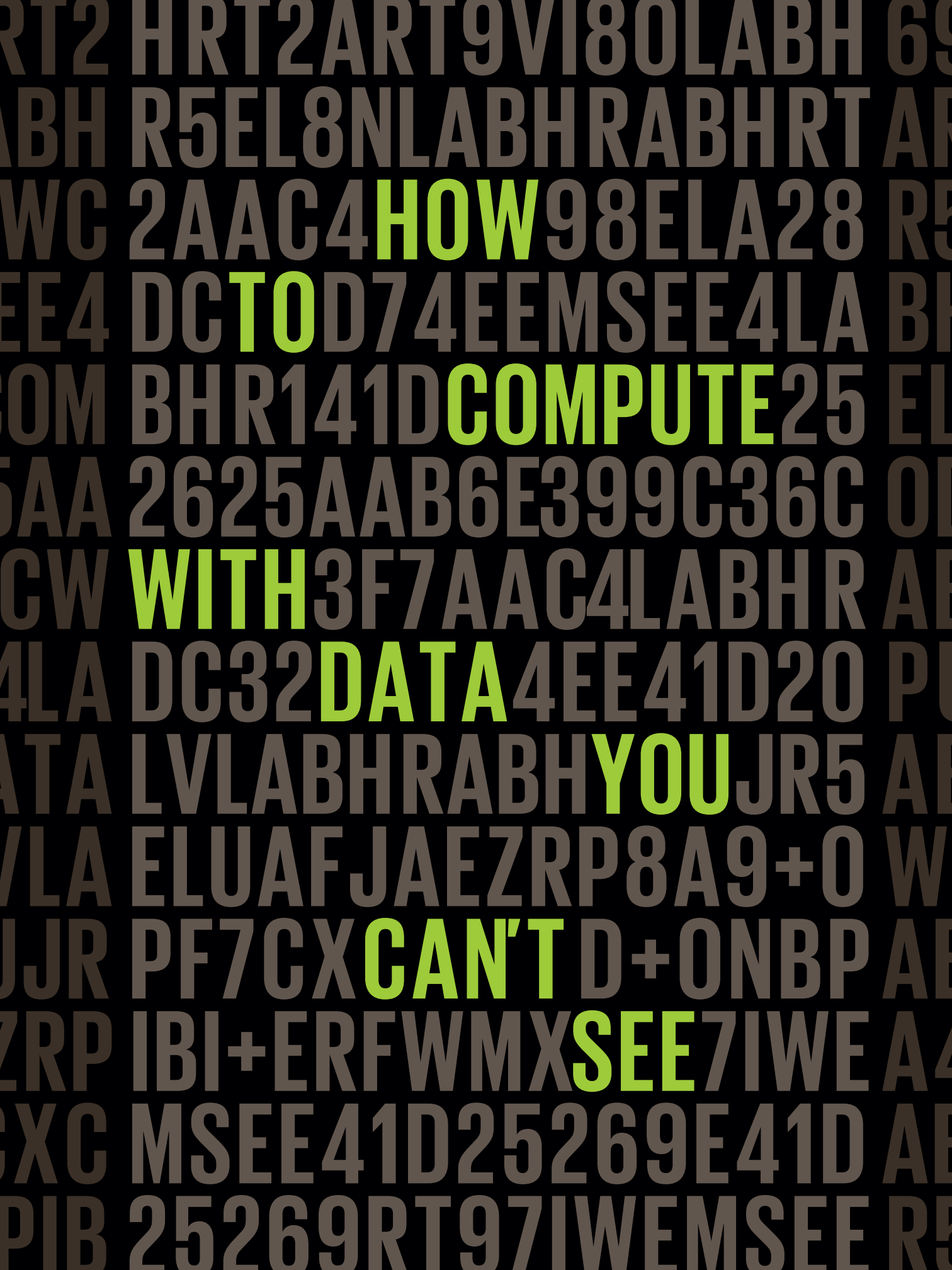

HOW
TO
COMPUTE
WITH
DATA
YOU
CAN'T
SEE

Web applications could increase security by keeping data encrypted even during computations

By Raluca Ada Popa & Nickolai Zeldovich

Not long ago, hackers stole about 40 million debit- and credit-card records from Target, another 56 million records from Home Depot, and nearly 5 million patient records from hospital operator Community Health Systems. And this past June, personal information about millions of federal employees was taken from the U.S. Office of Personnel Management. These are just a few thunderclaps in the perfect storm of cyberattacks and data breaches making headlines recently.

Despite massive efforts to guard sensitive data, hackers often manage to steal it anyway. It's a problem that's becoming especially acute, now that huge amounts of information are being concentrated on the servers of various cloud service providers. Most times we don't even know where these machines are located; how can we possibly feel that our data is safe with them?

Here's one way: Encrypt the data before it's stored. That way, even if attackers manage to break into the cloud provider's system and steal data, they'll just get meaningless gibberish.

This might seem a simple solution, but it has a big shortcoming: When data is encrypted, it's useless to the bad guys, for sure. But in many instances encryption makes it useless to the good guys as well.

Today's cloud providers typically perform many different kinds of useful computations on the data you entrust them with—looking things up, compiling statistics, analyzing trends, and so forth. Some apply very sophisticated machine-learning techniques to your data. But no one can do any of that if the data is encrypted.

How, after all, could Facebook possibly run a face-detection algorithm on your photos to recognize your friends if the images it holds are scrambled? And how could Amazon offer recommendations if it can't make sense of the purchase history it keeps on you?

So it would seem foolhardy to pursue encryption for anything other than perhaps simple data storage. In the past few years, however, a technique has emerged that achieves the seemingly impossible: It enables a cloud provider to perform many kinds of computations on data that has been encrypted.

The technique relies on special mathematical properties of certain encryption schemes that allow the cloud provider to carry out useful computations and produce an encrypted result. The end user can then decrypt that result to get the answer he or she is looking for.

The beauty of this approach is that the data stored by the cloud provider is always encrypted.

So even if someone steals every last bit of data from a cloud service's machines (or subpoenas the information on them), he gets only the encrypted data, which is essentially worthless. Indeed, this approach also protects you from the possibility of a hacker obtaining complete access to the cloud provider's computers and running the software on them at will. Even that intrusion won't reveal your data.

Today's work on computing with encrypted data has deep roots, reaching back almost four decades. But such computations are only now becoming practical, thanks in part to software tools we helped develop at the MIT Computer Science and Artificial Intelligence Laboratory. Those tools are complicated, but you don't need to understand their mathematical intricacies to get a general sense of how they work and how, contrary to intuition, you can compute useful things with data that you can't even see.

To make clear what we've been working on, consider a hypothetical example. Imagine that someone—we'll call her Alice—uses a medical Web application that runs in the cloud. She uses her browser to enter various kinds of sensitive information—disease symptoms, physical activity, diet, credit-card information for payments, and so forth—on the provider's website. But the company running this service is scrupulous about security. It has arranged things so that Alice's personal information gets encrypted on her local machine and is then sent to the cloud provider, so that the provider receives and stores only encrypted data.

Later, Alice asks for certain things to be computed based on what she had entered earlier—fitness level, diet recommendations, whatever. Remarkably enough, the cloud service can carry out these computations using just Alice's encrypted data. The answers will not seem to make any sense to the provider, but software running on Alice's machine automatically decrypts these results and presents them in Alice's browser in the usual way. So from her standpoint, the interaction with the Web application appears perfectly ordinary.

Let's imagine also that a doctor who is authorized to access the service asks for statistics about how many patients were sick in a given week, what the risk factors were for people who contracted a certain disease, or some other information. This doctor, too, gets results that are computed with encrypted data and returned to her machine in an encrypted form, at which point they are automatically decrypted and displayed.

In all, Alice and the doctor enjoy the same level of service they would have experienced with a regular Web application. The difference is that sensitive information is never exposed to hackers who might try to break into the provider's database or listen in on network communications.

How can this possibly work? For concreteness, let's consider a very simple computation. Imagine that the doctor wants to know the total number of people using the system who suffered from a specific disease during the past year. Assume that the cloud service has records of the number of people who reported this disease in each month of the year, but it holds that information in encrypted form.

To answer the doctor's query, the cloud provider needs to somehow add up 12 different encrypted numbers and return the result. That might seem impossible, but it can be done if the encryption scheme is chosen properly.

Conveniently, in 1999, while working on his thesis at École Nationale Supérieure des Télécommunications of Paris, Pascal Paillier developed an encryption system with a fantastic property: If you multiply a set of numbers after they've been encrypted, you will obtain, remarkably enough, the encrypted version of their sum.

So the cloud service in our example just needs to use Paillier's encryption system and multiply together the 12 encrypted values corresponding to the disease totals for each month of the year. This operation will generate

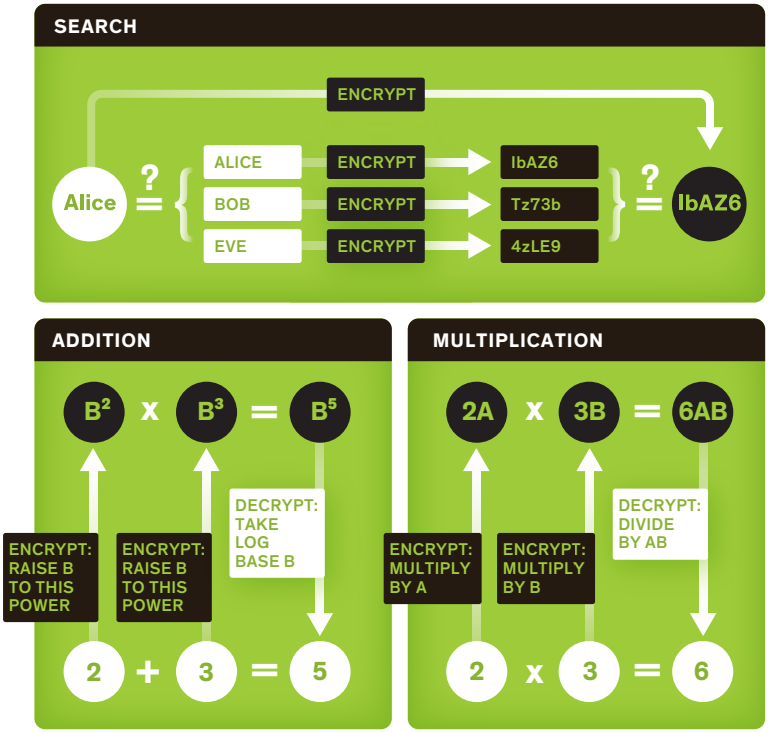
TOP 10 CORPORATE HACKS

THESE MASSIVE BREACHES go back more than a decade. An AOL employee stole 92 million records in 2004. The most recent victim here was Anthem, this past February.

Source: Information Is Beautiful (http://www.informationisbeautiful.net)

Company [industry]	Number of records stolen
eBay [e-commerce]	**145,000,000**
Heartland [financial]	**130,000,000**
T.J. Maxx / T.K. Maxx [retail]	**94,000,000**
AOL [Web]	**92,000,000**
Anthem [health care]	**80,000,000**
Sony [gaming]	**77,000,000**
JPMorgan Chase [financial]	**76,000,000**
Target [retail]	**70,000,000**
Home Depot [retail]	**56,000,000**
Evernote [Web]	**50,000,000**

MULTIPLE ENCRYPTION SCHEMES



FOR SIMPLE SEARCHING [top], any encryption scheme will work, but for other operations you need to choose an appropriate method. To perform addition, for example, you could use exponentiation to encrypt the data, multiply the results, and compute the logarithm to decrypt [left]. For multiplication, you could first multiply by some chosen constants to encrypt the two numbers, multiply the encrypted values, and then divide by the product of the constants to decrypt the result [right]. These simple encryption schemes would not, of course, be sufficiently secure to use in practice, but they show the general strategy.

a value that is the encryption of the sum of those monthly tallies, without the service ever having access to the individual values. The cloud service returns this result to the doctor's machine, which decrypts the value and displays the total for the year on her computer screen.

This general approach to working with encrypted data isn't limited to simple addition. There are all kinds of other things you can do with encrypted data if you pick the right encryption scheme, including comparison, sorting, multiplication and other arithmetic operations, as well as trigonometric functions.

The idea of computing with encrypted data arose first in 1978, when Ronald Rivest, Len Adleman, and Michael Dertouzos wrote a seminal article titled "On Data Banks and Privacy Homomorphisms." In it, they introduced the idea of keeping data encrypted while computing things with it. They called an encryption scheme that could support such computation "homomorphic." They did not know at the time how to carry out an encryption that would allow all sorts of computations to be performed—and neither did anybody else—but they and other computer scientists were eager to find a way.

The quest lasted for more than 30 years. In 2009, Craig Gentry, then a graduate student at Stanford University, made a major breakthrough: He came up with an encryption scheme that allows a computer to calculate any function at all on data after it has been encrypted. Such a scheme is called fully homomorphic encryption.

From a mathematical standpoint, Gentry's solution to the problem was truly beautiful. And soon, other researchers proposed additional fully homomorphic encryption systems aimed primarily at improving the performance and security of Gentry's original scheme.

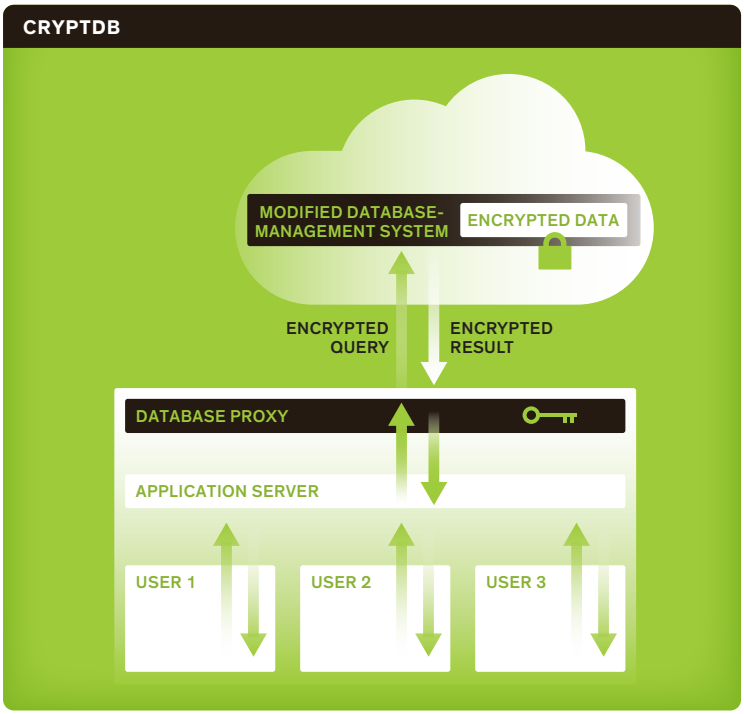
Despite this progress, a huge problem remained: The best fully homomorphic encryption schemes took more than a million times as long to complete as the corresponding unencrypted computations. If it normally took a second for a website to compute your results, with fully homomorphic encryption you'd have to wait about 12 days. Such sluggishness was clearly a showstopper.

Then in 2011 a team of security and cryptography experts, which included the two of us, built a system called CryptDB. It allowed a Web application to perform a range of database queries in the widely used Structured Query Language (SQL) with only a 27 percent performance slowdown.

What was the trick? The key was to get away from the idea that one encryp-

INFOGRAPHIC BY Carl De Torres

SAFETY AT ALTITUDE



SECURITY IS OFTEN a concern when information is held in a cloud database. CryptDB addresses this issue by encrypting the data in a way that still allows normal database queries to be performed. The application itself runs locally, as does a database proxy, which performs the encryption and decryption. The proxy also translates queries into a form that can be run on the modified database-management system running in the cloud.

tion system would work for everything. Fully homomorphic encryption aims to support all functions within a single encryption scheme. That makes it slow even for simple operations.

We and our colleagues realized that an encryption scheme specialized for just one operation on the encrypted data could be much faster. Paillier's encryption scheme, for example, can compute the sum of encrypted values very quickly, but it can't compute anything else.

To support a variety of operations, then, you need to use a variety of specialized encryption schemes. Each is efficient at just one thing, but together they cover quite a lot of territory.

Currently, there are specialized (and fast) algorithms for many common operations, some of which we developed: addition, multiplication, comparison by equality or by order, set intersection, polynomial computation, machine-learning classification tasks, searching encrypted text, and others. Using all of them to encrypt your data, and therefore storing multiple sets of encrypted data, allows you to perform a variety of different computations with the encrypted results. You just switch back and forth among the encrypted data sets, in each instance using the one that corresponds to the operation you need done.

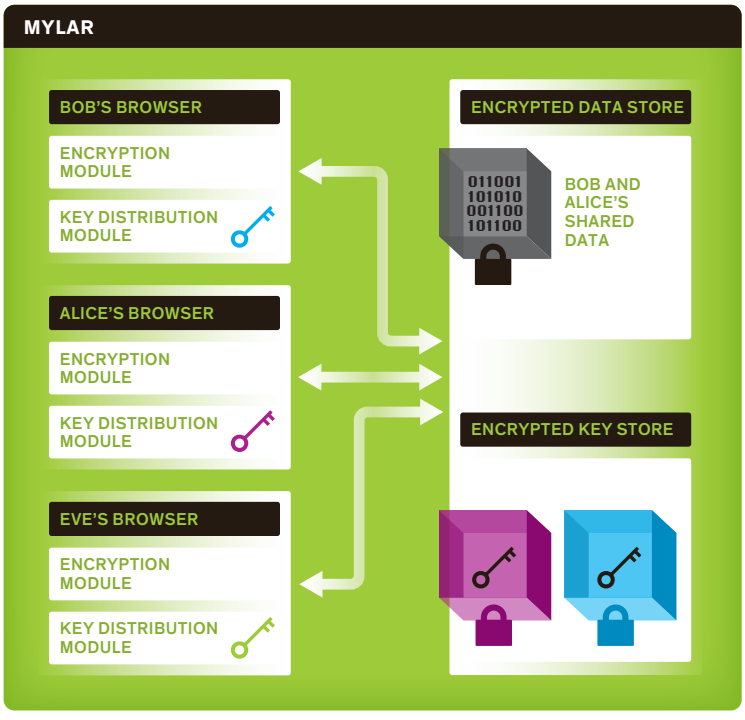
CryptDB exploited this insight for the first time in a practical way. As a result, it has gained traction in industry. For example, following CryptDB's lead (and giving credit to it), Google recently deployed a system called Encrypted BigQuery. It can perform queries on an encrypted version of Google's BigQuery database. And the software giant SAP implemented a system called Search Over Encrypted Data, which uses CryptDB on top of SAP's High-Performance Analytic Appliance database server. Also, researchers at MIT's Lincoln Laboratory use CryptDB for a special version of the open-source Apache Accumulo database.

Because each of these systems uses a variety of different encryption schemes, application designers are limited in how they can combine different operations. Still, being able to do SQL queries on a collection of encrypted data is often all you need.

Last year, we developed a system called Mylar to add to the capabilities of CryptDB. Mylar goes beyond just querying a database full of encrypted data—it enables users of a Web application to also share data with one another. Such data sharing is a staple of many Web applications: Facebook users share photos and posts with one another, users of an online calendar share events, and so forth. Mylar enables all such sharing, according to whatever permissions the user grants others.

INFOGRAPHICS BY Carl De Torres

SHARING WHILE CARING



THE AUTHORS' MYLAR Web-application framework allows users to share data with those they choose. This is done by encrypting shared data with particular encryption keys and then storing encrypted versions of those keys in such a way that only the proper users can access them. Here Bob and Alice use their keys [blue, purple] to obtain copies of a third key [black], which they each can use to access their shared encrypted data.

For instance, let's say Alice of our hypothetical example wants to share her medical history with her doctor so that she can be treated. The desire here is that both Alice and her doctor be able to decrypt Alice's medical information. A hacker, whom we'll call Malice, shouldn't be able to decrypt Alice's data, even if Malice manages to hack the cloud-service provider and steal all the data and code stored there.

The same should apply, of course, to any number of people who've chosen to share their information with this doctor—perhaps it's everyone in the database. Indeed, allowing a doctor access to everyone's data would be a prerequisite for answering many important questions. For instance, the doctor might want to search all the data stored on the cloud medical application to look for people with a rare disease. The search request she sends to the Web application would contain an encrypted keyword corresponding to the name of that disease. CryptDB could handle that request if all the data were encrypted with the same cryptographic key. The problem, of course, is that different people's records will inevitably be encrypted with different keys, so searching through the whole set is normally impossible.

Mylar skirts the problem by distributing a shared encryption key to users who want to share data. That must be done carefully, to prevent a hacker from tricking users into sharing data with a server he controls.

To avoid that, Mylar includes a special browser extension that verifies the code downloaded from the server. The system still works without it, but less securely. Mylar also offers an identity-provider service, which acts like a Web certificate authority. (A Web certificate authority is an entity that helps ensure that you are connecting with the real thing when you visit a website using https, the secure version of hypertext transfer protocol.)

We have used Mylar to secure a variety of Web applications—for health care, chats, forums, photo sharing, calendars, and online courses. These experiments showed that Mylar is fast: It increases computation time by only 17 percent on average.

Mylar has also been adopted in a real-world application, one used at the Newton-Wellesley Hospital, in Newton, Mass., to collect information about women with endometriosis, a painful abdominal disorder. At this very moment, Mylar is helping to protect the privacy of these patients.

We are confident that computing with encrypted data, using systems like CryptDB and Mylar, will become one of the primary strategies for protecting confidential information stored in the cloud. And this approach can protect more than just data: It's also been used to secure cloud computers running linear algebraic operations, big-data analytics, and machine-learning tools.

The security of information stored online is a huge problem these days, and computing on encrypted data could be an important part of the solution. It protects sensitive information against theft for the simple reason that if even the company holding the data has no idea what the values mean, an attacker will have nothing of value to steal. ■

POST YOUR COMMENTS at http://spectrum.ieee.org/computation0815