Core Concept: Homomorphic encryption

Robert Frederick
Science Writer

It turned out to be pretty easy for Yaniv Erlich to identify people who had donated their genomic data to science, even despite researchers' attempts to make that data anonymous. Erlich, a former computer hacker hired by businesses to test the security of computer systems, only needed his tailor-made computer program and an Internet connection to access publicly available databases.

Erlich, though, had no nefarious intentions. As a fellow at the Whitehead Institute for Biomedical Research, his was an academic exercise to show that such privacy systems were flawed. To keep others from repeating his work, however, Erlich omitted some of the necessary steps from his group's published paper (1). Although this was an unusual omission for a scientific paper, Erlich was concerned that someone could repeat his work and reveal the individuals' identities, and expose their genomic data—potentially affecting relationships, employability, and insurability.

Erlich says the success rate of his technique is about 12% because he was only looking at a partial sequence of men's Y chromosomes, plus their ages, and where they lived in the United States. But he notes that given growing amounts of genomic information, more and more people's identities could be revealed. The ways in which Erlich and others have found to uncover individuals' identities have left institutions little choice but to remove previously published genomic data from the Internet (2) and impose strict limitations on data use (3).

Should it prove feasible, homomorphic encryption could provide much needed security for all sorts of sensitive shared data. Image courtesy of Shutterstock/Lightspring.

Genomics is one of many fields that would greatly benefit from more secure ways to store and share data. An approach called "homomorphic encryption," still being developed, may help.

Homomorphic encryption allows people to use data in computations even while that data are still encrypted. This just isn't possible with standard encryption methods. The method is called "homomorphic" (or "same form") encryption because the transformation has the same effect on both the unencrypted and encrypted data. For example, suppose an encryption scheme entailed multiplying numbers by 10 and the decryption entailed dividing them by 10. This encryption is homomorphic for simple addition because 2 + 3 would be encrypted to 20 + 30, and decrypting the answer by dividing by 10 would get to 5, as expected. A standard encryption method, though, might turn a 2 into a smiley face and a 3 into a semicolon. Adding such symbols is nonsensical, making calculations impossible.

But there are different degrees of homomorphic encryption, sometimes referred to as "fully" homomorphic compared with "partly." The above encryption scheme, for example, is only partly homomorphic because it does not work for multiplication. Multiplying 2 by 3 would be encrypted as 20*30, and decrypting the answer, 600, gets you 60, not 6, as desired.

Making an encryption scheme fully homomorphic is conceptually straightforward; in the above example, doing so means defining "encrypted multiplication" to include dividing by 10. Making a fully homomorphic encryption scheme secure, however, has been the hard part.

When secure fully homomorphic encryption was first introduced as a concept in the literature in 1978 by researchers at the Massachusetts Institute of Technology (4), it wasn't yet known whether it was even possible. Not until 2009 did Craig Gentry, then a graduate student at Stanford, find a way to do it (5). But that solution, says Kristin Lauter, research manager of Microsoft Research's cryptography group, was "literally not implementable." For example, Gentry himself estimated a simple encrypted Google search using his original method would take roughly a trillion times longer than a typical search. As a result, a one-second unencrypted search would take 31,688 years when encrypted.

By 2013, however, Gentry had moved to IBM Research, where his colleagues had developed a secure version of fully homomorphic encryption that only took a million times longer than a typical search. A one-second unencrypted search would then take around 12 days encrypted. By Gentry's estimates, efficient fully homomorphic encryption could be ready in another decade. Others, such as Bruce Schneier, chief technology officer at Resilient Systems, are not so hopeful, however; he estimates 40 years or more (6).

By 2011, though, Lauter and her colleagues had a proof-of-concept implementation of a "somewhat" homomorphic encryption scheme (7). Instead of being the one-size-fits-all solution of fully homomorphic encryption, Lauter's team's approach depends on specifying parameters in advance, including what computation is needed, dataset size, and data ranges. For example, suppose one wanted to use Lauter's "somewhat" scheme to securely calculate the heart attack risk for all of the people in the Framingham Heart Study. This might require specifying the number of people in the study, the algorithm for computing heart attack risk, the parameters used in performing the calculation (e.g., weight and age), and limits to the size of the data parameters (setting age, say, between 1 and 150 years and weight between 1 and 300 kilograms).

Lauter's system, she claims, can be understood by someone familiar with the mathematical tools most students pick up in undergraduate mathematics courses or when pursuing an MBA. "Simplicity is an advantage when you're trying to deploy and maintain secure systems," Lauter says.

In 2014, her team published (8) a paper on what she now calls "practical homomorphic encryption," requiring parameters set in advance. Microsoft Research hasn't made the system available yet. But Lauter says that even after the release there will be a "long process for the scientific and business communities to accept new encryption techniques, understand the potential impact, and deploy them." Other companies are developing their own versions of practical homomorphic encryption, but so far they either are not releasing details or their systems are for researchers working on homomorphic encryption schemes, such as with IBM's release of its homomorphic encryption software library, HElib (https://github.com/shaih/HElib).

In the coming years, though, homomorphic encryption could have a big role to play if, as Gentry writes, the method makes it possible "to delegate *processing* of your data without giving away *access* to it." (9) Already, cryptography researchers are holding competitions to challenge the various homomorphic encryption schemes. One of those competitions (www.humangenomeprivacy.org/2015/competition-tasks.html) even asks participants to use homomorphic encryption to securely share the most personal data people have: their genomes.

1 Gymrek M, McGuire AL, Golan D, Halperin E, Erlich Y (2013) Identifying personal genomes by surname inference. *Science* 339(6117):321–324.
2 Landry JJ, et al. (2013) The genomic and transcriptomic landscape of a HeLa cell line. *G3 (Bethesda)* 3(8):1213–1224.
3 Paltoo DN, et al. (2014) Data use under the NIH GWAS data sharing policy and future directions. *Nat Genet* 46(9):934–938.
4 Rivest R, Adleman L, Dertouzos M (1978) On data banks and privacy homomorphisms. *Foundations of Secure Computation*. eds DeMillo R, Dobkin D, Jones A, Lipton R, eds (Academic, New York), pp 169–180.
5 Gentry C (2009) Fully homomorphic encryption using ideal lattices. *Proceedings of the 41st Annual ACM Symposium on the Theory of Computing* (ACM, New York), pp 169–178.
6 Schneier B (2009) Homomorphic encryption breakthrough. Available at https://www.schneier.com/blog/archives/2009/07/homomorphic_enc.html. Accessed May 19, 2015.
7 Lauter K, Naehrig M (2011) Can homomorphic encryption be practical? *Proceedings of the 3rd ACM Cloud Computing Security Workshop* (ACM, New York), pp 113–124.
8 Lauter K, Boss JW, Naehrig M (2014) Private predictive analysis on encrypted medical data. *J Biomed Inform* 50:234–243.
9 Gentry C (2010) Computing arbitrary functions of encrypted data. *Communications of the ACM* 53(3):97–105. Available at crypto.stanford.edu/craig/easy-fhe.pdf. Accessed May 19, 2015.