IBM completes successful field trials on Fully Homomorphic Encryption

Enlarge / We're already accustomed to data being encrypted while at rest or in flight—FHE offers the possibility of doing computations on it as well, without ever actually decrypting it.

reader comments

37 with 27 posters participating

Share this story

Yesterday, Ars spoke with IBM Senior Research Scientist Flavio Bergamaschi about the company's recent successful field trials of Fully Homomorphic Encryption. We suspect many of you will have the same questions that we did—beginning with "what *is* Fully Homomorphic Encryption?"

FHE is a type of encryption that allows direct mathematical operations on the encrypted data. Upon decryption, the results will be correct. For example, you might encrypt 2, 3, and 7 and send the three encrypted values to a third party. If you then ask the third party to add the first and second values, then multiply the result by the third value and return the result to you, you can then decrypt that result—and get 35.

You don't ever have to share a key with the third party doing the computation; the data remains encrypted with a key the third party never received. So, while the third party performed the operations you asked it to, it never knew the values of either the inputs or the output. You can also ask

the third party to perform mathematical or logical operations of the encrypted data with *non*-encrypted data—for example, in pseudocode, FHE_decrypt(FHE_encrypt(2) * 5) equals 10.

Homomorphic Encryption possibilities

Enlarge / The most obvious application of FHE is solving what I like to call “the sysadmin problem”—preventing secret discovery by root-privileged operators.

The most obvious implication of FHE is a solution to what I like to call “the sysadmin problem”—if you do your computation on a system managed by a third party, the root-privileged operators at the third party generally have access to the data. Encryption at rest prevents access to the data outside the scope of whatever computation is going on at that specific moment—but with root privileges, a system operator can scan or alter the contents of RAM to gain access to whatever data is currently being operated on.

With FHE, you can perform those calculations without the actual data ever being exposed to the remote system at all. Obviously, this solves the sysadmin problem pretty

thoroughly—when the machine itself never has access to the decrypted data, neither do its operators.

Of course, FHE isn't the first solution to the sysadmin problem—AMD's Secure Encrypted Virtualization is another, and it's considerably more efficient. With SEV enabled, an operator who has root privilege on a host system can't inspect or meaningfully alter the contents of RAM in use by a virtual machine running on that system. SEV is, effectively, free—SEV-protected VMs don't operate any more slowly than non-protected VMs do.

Enlarge / It's hard to come up with all the possibilities of fully homomorphic encryption off the top of your head—so here are a few to get us started.

Fully Homomorphic Encryption offers many possibilities that Secure Encrypted Virtualization does not, however. Since all mathematical and logical operations can be built from additive and multiplicative operations, this effectively means that any computation can be performed upon FHE encrypted data. This opens a dizzying array of possibilities: one might search a database without ever letting the database owner know what you searched for or what the result was. Two parties might discover the intersection set of their separately held

datasets without either party revealing the actual contents of their data to the other.

“Secure outsourcing” is the only one of these archetypes that is possible without Fully Homomorphic Encryption. It’s worth spending most of our time focusing on the other three, which do require it—because FHE comes at a significant cost.

Homomorphic Encryption limitations

- This is a multiplicative scale, not a percentage scale—you’re looking at, in many cases, 42x the computational power and 10-20x the memory needed to perform ML operations on FHE encrypted data.
- The true/false positive prediction curves on the left are similar, but not identical—the FHE encryption itself isn’t lossy, but it’s operating on floating point data, which is.
- This diagram of a cloud-based machine-learning and inference setup demonstrates how both storage and computation can be done on private models by untrusted third parties.

Although Fully Homomorphic Encryption makes things possible that otherwise would not be, it comes at a steep cost. Above, we can see charts indicating the additional compute power and memory resources required to operate on FHE-encrypted machine-learning models—roughly 40 to 50 times the compute and 10 to 20 times the RAM that would be required to do the same work on unencrypted models.

In the next image, we can see that the result curves of a machine-learning prediction task are very nearly identical whether the operations were done on data in the clear or on FHE-encrypted data. We wondered about the remaining difference—was FHE a little bit lossy? Not exactly, Bergamaschi explained. The model in use is based on

floating-point data, not integer—and it's the floats themselves that are a little lossy, not the encryption.

Each operation performed on a floating-point value decreases its accuracy a little bit—a very small amount for additive operations, and a larger one for multiplicative. Since the FHE encryption and decryption themselves are mathematical operations, this adds a small amount of additional degradation to the accuracy of the floating-point values.

We should stress that these charts are only directly applicable to machine learning, and not every task lending itself to FHE is a machine-learning task. However, other tasks present their own limitations—for example, we spent some time going back and forth over how a blind search (one in which the search operator knows neither what you searched for nor the result it gave you) could work.

Normally, if you query a database, the database doesn't need to do a full text search on every row in the table(s) being queried—the table(s) will be indexed, and your search can be tremendously accelerated by use of those indices. If you're running a blind search using an FHE-encrypted value, however, your encrypted query must be masked against every full-text row in the queried table(s).

In this way, you can both submit your query and get your result without the database operator possessing knowledge of either—but that full-text read *and* masking operation against every single row in the queried tables won't scale very well, to say the least. How much of a problem this is depends greatly on the type of data being queried and how much of it there is—but we're probably looking at machine learning's 50:1 compute penalty and 20:1 memory penalty all over again, if not worse.

Successful field trials

- IBM has completed two field trials of FHE with real data in the financial industry—one with a large American bank, and one with a large European bank.
- This diagram of the FHE trial with a large American bank demonstrates using machine-learning models against financial data to determine likelihood of a loan being issued.

As daunting as the performance penalties for FHE may be, they're well under the threshold for usefulness—Bergamaschi told us that IBM initially estimated that the minimum efficiency to make FHE useful in the real world would be on the order of 1,000:1. With penalties well under 100:1, IBM contracted with one large American bank and one large European bank to perform real-world field trials of FHE techniques, using live data.

The American trial involved was based upon using machine-learning models to predict the likelihood of a loan being issued and given access to retail banking, loan, and investment information for a large group of customers. The dataset consisted of 364,000 entries, with several hundred features—and the model needed to identify relatively rare events (about 1 percent) in the dataset. Success for the trial was based on the encrypted predictions having similar accuracy to baseline, variable selection with similar accuracy, and acceptable compute overhead.

We don't have as much detail about the European study, as it's still under NDA—but the results of the American study were published in 2019 and can be viewed in detail at the Cryptology ePrint archive.

IBM's Homomorphic Encryption algorithms use lattice-based encryption, are significantly quantum-computing resistant, and

are available as open source libraries for Linux, MacOS, and iOS. Support for Android is on its way.

https://arstechnica.com/?p=1695410