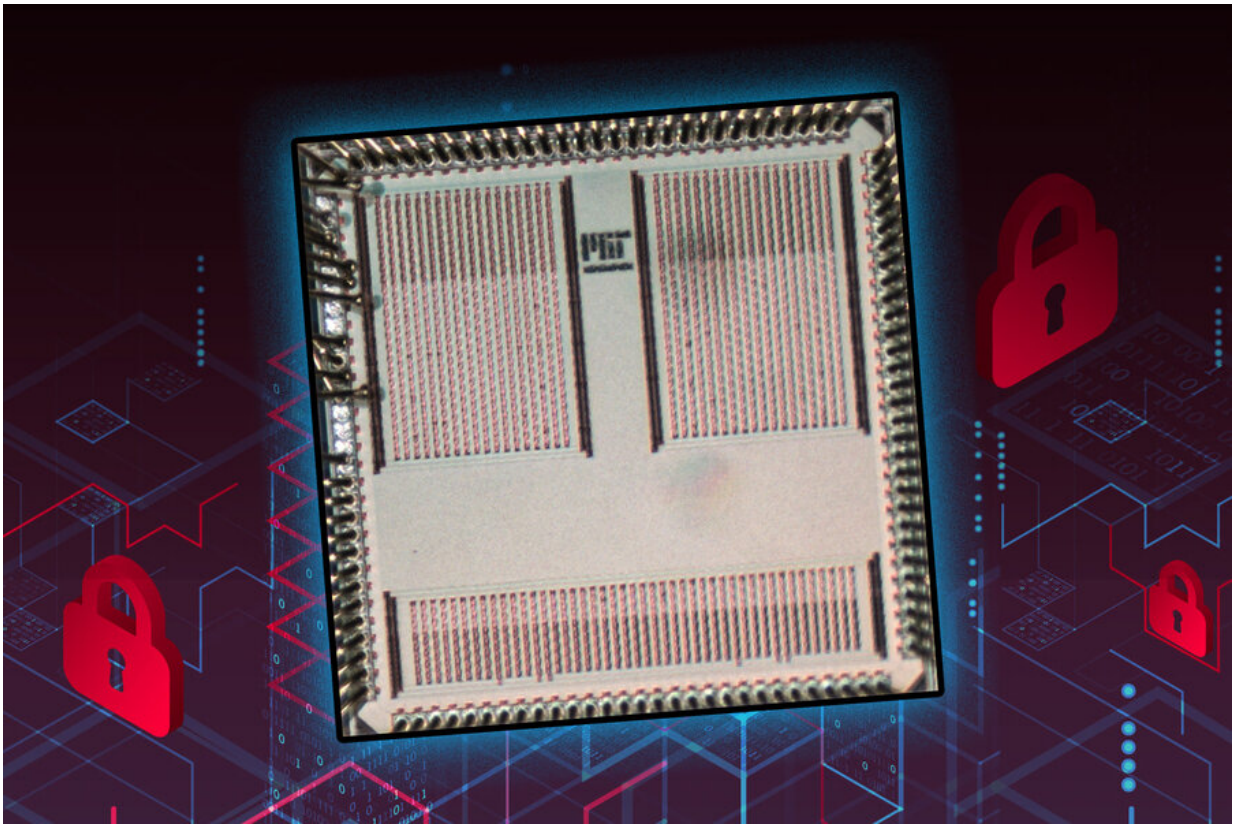
A security technique to fool would-be cyber attackers

February 23 2022, by Adam Zewe



MIT researchers developed an application-specific integrated circuit (ASIC) chip, pictured here, that can be implemented on an internet of things device to defend against power-based side-channel attacks. Credit: Massachusetts Institute of Technology

Multiple programs running on the same computer may not be able to

directly access each other's hidden information, but because they share the same memory hardware, their secrets could be stolen by a malicious program through a "memory timing side-channel attack."

This malicious program notices delays when it tries to access a computer's memory, because the hardware is shared among all programs using the machine. It can then interpret those delays to obtain another program's secrets, like a password or cryptographic key.

One way to prevent these types of attacks is to allow only one program to use the memory controller at a time, but this dramatically slows down computation. Instead, a team of MIT researchers has devised a new approach that allows memory sharing to continue while providing strong security against this type of side-channel attack. Their method is able to speed up programs by 12 percent when compared to state-of-the-art security schemes.

In addition to providing better security while enabling faster computation, the technique could be applied to a range of different side-channel attacks that target shared computing resources, the researchers say.

"Nowadays, it is very common to share a computer with others, especially if you are do computation in the cloud or even on your own mobile device. A lot of this resource sharing is happening. Through these shared resources, an attacker can seek out even very fine-grained information," says senior author Mengjia Yan, the Homer A. Burnell Career Development Assistant Professor of Electrical Engineering and Computer Science (EECS) and a member of the Computer Science and Artificial Intelligence Laboratory (CSAIL).

The co-lead authors are CSAIL graduate students Peter Deutsch and Yuheng Yang. Additional co-authors include Joel Emer, a professor of

the practice in EECS, and CSAIL graduate students Thomas Bourgeat and Jules Drean. The research will be presented at the International Conference on Architectural Support for Programming Languages and Operating Systems.

Committed to memory

One can think about a computer's memory as a library, and the memory controller as the library door. A program needs to go to the library to retrieve some stored information, so that program opens the library door very briefly to go inside.

There are several ways a malicious program can exploit shared memory to access secret information. This work focuses on a contention attack, in which an attacker needs to determine the exact instant when the victim program is going through the library door. The attacker does that by trying to use the door at the same time.

"The attacker is poking at the memory controller, the library door, to say, 'is it busy now?" If they get blocked because the library door is opening already—because the victim program is already using the memory controller—they are going to get delayed. Noticing that delay is the information that is being leaked," says Emer.

To prevent contention attacks, the researchers developed a scheme that "shapes" a program's memory requests into a predefined pattern that is independent of when the program actually needs to use the memory controller. Before a program can access the memory controller, and before it could interfere with another program's memory request, it must go through a "request shaper" that uses a graph structure to process requests and send them to the memory controller on a fixed schedule. This type of graph is known as a directed acyclic graph (DAG), and the team's security scheme is called DAGguise.

Fooling an attacker

Using that rigid schedule, sometimes DAGguise will delay a program's request until the next time it is permitted to access memory (according to the fixed schedule), or sometimes it will submit a fake request if the program does not need to access memory at the next schedule interval.

"Sometimes the program will have to wait an extra day to go to the library and sometimes it will go when it didn't really need to. But by doing this very structured pattern, you are able to hide from the attacker what you are actually doing. These delays and these fake requests are what ensures security," Deutsch says.

DAGguise represents a program's memory access requests as a graph, where each request is stored in a "node," and the "edges" that connect the nodes are time dependencies between requests. (Request A must be completed before request B.) The edges between the nodes—the time between each request—are fixed.

A program can submit a memory request to DAGguise whenever it needs to, and DAGguise will adjust the timing of that request to always ensure security. No matter how long it takes to process a memory request, the attacker can only see when the request is actually sent to the controller, which happens on a fixed schedule.

This graph structure enables the memory controller to be dynamically shared. DAGguise can adapt if there are many programs trying to use memory at once and adjust the fixed schedule accordingly, which enables a more efficient use of the shared memory hardware while still maintaining security.

A performance boost

The researchers tested DAGguise by simulating how it would perform in an actual implementation. They constantly sent signals to the memory controller, which is how an attacker would try to determine another program's memory access patterns. They formally verified that, with any possible attempt, no private data were leaked.

Then they used a simulated computer to see how their system could improve performance, compared to other security approaches.

"When you add these security features, you are going to slow down compared to a normal execution. You are going to pay for this in performance," Deutsch explains.

While their method was slower than a baseline insecure implementation, when compared to other security schemes, DAGguise led to a 12 percent increase in performance.

With these encouraging results in hand, the researchers want to apply their approach to other computational structures that are shared between programs, such as on-chip networks. They are also interested in using DAGguise to quantify how threatening certain types of side-channel attacks might be, in an effort to better understand performance and security tradeoffs, Deutsch says.

This story is republished courtesy of MIT News (web.mit.edu/newsoffice/), a popular site that covers news about MIT research, innovation and teaching.

Provided by Massachusetts Institute of Technology