

UMass Researchers Describe New Approach to Tag Security

The scientists seek to leverage a unique signal generated by an RFID chip's volatile memory so the tag can be authenticated and its data encrypted.

By Mary Catherine O'Connor

Nov. 1, 2007—Passive RFID tags made with chips containing volatile memory are a bit like snowflakes, according to researchers from the [University of Massachusetts Amherst \(UMass\)](#): each one is unique. The researchers say they've identified methods of exploiting the tags' uniqueness so that the tags can be authenticated (to deter tag cloning) and its data secured with cryptography—something that can not currently be done with most passive RFID tags.

The UMass research is receiving financial support from the [RFID Consortium for Security and Privacy \(CUSP\)](#), a research initiative funded by a \$1.1 million grant from the [National Science Foundation](#) (see [RFID Security Consortium Receives \\$1.1 Million NSF Grant](#)).

An RFID tag's memory is composed of memory cells, each one a circuit representing a single bit of data. When an RFID interrogator's RF signal powers up the tag, the chip's memory cells produce a unique fluctuating electrical pattern before generating the ID and any other data encoded to the chip (and transmitted by the tag in the form of a modulated RF signal).

Prior to sending its ID number, a tag emits an RF signal encoded with this unique electrical pattern. The UMass researchers have shown that this pattern, when captured by an RFID interrogator, can serve as a unique fingerprint that can then be saved and used to authenticate the tag the next time it is read. It could also be used to ensure, for example, that only legitimate RFID-tagged products are exchanged between trading partners.

In addition, a tag's fingerprint could be used to encrypt the tag's encoded data, thereby securing it, explains Daniel Holcomb. A former graduate student at UMass, and one of the drivers of the research, Holcomb is now pursuing his Ph.D. at the University of California, Berkeley. According to Holcomb, an end user could apply a tag's fingerprint to a randomness extractor that is part of a hash cryptography process. This would create a string of unique, random numbers that could be used to generate keys to decrypt the stored tag data. A party looking only to collect the encrypted data on a tag could use any reader. To use or change the data, that party would need specialized software to generate the keys and use them to decrypt the data.

Holcomb says the researchers have not conducted any real-world tests to determine if their theory—that a tag's fingerprint could be used in this manner—is viable. However, he adds, key to this research is the fact that the authentication and security features could be implemented on a passive RFID tag without adding as much additional memory or circuitry to support it as would be required for traditional data security on an RFID tag, because the fingerprint is already generated each time a tag is read.

RELATED_ARTICLES One significant inhibitor to applying these authentication and encryption methods to passive UHF tags, Holcomb indicates, is that they require the chip to be inside the tag and use volatile, static random access memory (SRAM). Although some types of RFID tags use SRAM chips, EPC Gen 2 tags (the type used widely in supply chain applications for tracking and tracing products) employ nonvolatile, electrically erasable, programmable, read-only memory (EEPROM) chips. Nonvolatile memory, Holcomb says, is less expensive, requires a smaller chip than volatile memory and is used to keep Gen 2 tag costs low. But only volatile memory generates the unique electrical fluctuation that can serve as a fingerprint.

That doesn't mean this approach couldn't work on Gen 2 or other UHF tags, he notes—just that the tags would need to use volatile memory. Holcomb and his colleagues demonstrated the feasibility of such an approach by utilizing it on an EPC Gen 1-compliant tag with volatile memory.

Copyright ©2005 RFID Journal, Inc. All Rights Reserved