# On Security with the New Gen2 RFID Security Framework

Daniel W. Engels[†], *Senior Member, IEEE,* You Sung Kang[‡], *Member, IEEE,* and Junyu Wang[*] *Member, IEEE*

***Abstract*—Radio frequency identification (RFID) systems compliant to the EPCglobal Generation 2 (Gen2) passive UHF RFID protocol are being deployed in a broad range of applications including access control, automated tolling, personal identification, anti-counterfeiting, and supply chain management. With the broad applications and the demand for ever increasing amounts of on-tag functionality, security on the tag has become a critical enabling functionality in many applications. To address this growing marketplace need, EPCglobal is developing a standard security framework within which security functionality may be integrated seamlessly into the Gen2 protocol. We review the proposed Gen2 security framework and introduce example cryptographic suites to illustrate how to utilize this framework to provide a range of security functionality. We analyze the security of the Gen2 protocol and this new functionality in the context of timing-based attacks. We conclude that the tight communication timings specified in the Gen2 protocol mitigate timing-based attacks; however, the loose timing implementations on commercial interrogators and limited timing enforcement on tags lesson the effectiveness of the specified timing constraints. Further, we conclude that the new security framework allows for the efficient integration of secure functionality that, as specified, is resistant to timing-based attacks; however, we caution that using the delayed response of the new Gen2 security functionality creates new vulnerabilities to timing based attacks such as relay attacks and man-in-the-middle attacks.**

## I. Introduction

THE EPCglobal Generation 2 (Gen2) passive UHF RFID (radio frequency identification) protocol has received significant commercial attention and adoption in a broad range of applications since its initial ratification in 2005. Development of the Gen2 protocol began in early 2003 under the direction of the MIT Auto-ID Center[1], and on June 11, 2003 Wal-Mart CIO Linda Dillman announced that Wal-Mart would require its top 100 suppliers to tag pallets and cases of goods with passive UHF RFID tags [1]. The US Department of Defense announced similar initiatives shortly after Wal-Mart's announcement. Both organizations indicated timelines and preference for the (at that time) future Gen2 products. With this potentially huge market, companies have invested heavily in evolving Gen2 systems from the low performance mid-range passive RFID systems of 2005 to high-performance long-range passive RFID systems that exist today.

The Gen2 protocol was designed for retail supply chain management with the minimalist on-tag functionality philosophy of the Auto-ID Center's EPC System [2][3]. The minimalist approach is well-suited to the functional and security requirements of the retail supply chain. The primary functionality requirements were for the tag to carry a unique identifier, the Electronic Product Code (EPC), to return this identifier during an efficient identification process, and to operate in a dense interrogator environment. The primary security requirements were to mitigate eavesdropping on the interrogator's communications, to allow for password protected access to tag user memory, and to provide for a password secured kill functionality. The resulting Gen2 standard protocol v1.0.9 [4] and its evolutionary brethren [5][6] meet all of their original design requirements, making them secure for and functionally useful within the retail supply chain environment.

Gen2 products have been adopted in applications far beyond the retail supply chain with applications including maintenance and repair operations, electronic vehicle registrations, anti-counterfeiting of goods, and identification cards such as driver's license. The ubiquity of Gen2 systems has increased the likelihood of attacks against these systems. Furthermore, the security requirements for some of these applications far exceeds the original security requirements for the Gen2 protocol. The security limitations of the Gen2 protocol have been well-documented. Prior to the Gen2 development, Sarma et.al. [3] identified several potential attacks in the consumer environment and proposed a hash-based mechanism to mitigate some of these attacks. The `KILL` command in the Gen2 protocol exists solely as a mechanism to mitigate attacks on consumers, although this works to deprive the consumer of the use of the tags. A body of work evaluating the security limitations of the Gen2 protocol and passive UHF RFID broadly and proposing security mechanisms and algorithms for low cost RFID has emerged since the original publication of the Gen2 specification. Some of these proposals work within the confines of the Gen2 protocol, most notably [7]; however, most proposals require the addition of custom commands. A summary of early work on this topic can be found in [8] with an analysis of the security deficiencies for a range of non-retail supply chain applications being evaluated in [9].

The demand for increased security functionality within the Gen2 protocol led EPCglobal and the International Organization for Standardization (ISO) together with the International Electrotechnical Commission (IEC) under Joint Technical Committee 1, Sub Committee 31, Work Group 7 (ISO/IEC

[†]D.W. Engels is a Visiting Fellow in the Computer Science and Engineering Department at Southern Methodist University, Dallas, Texas, USA (email: dwe@alum.mit.edu).

[‡]Y.S. Kang is with the Department of Electrical Engineering, KAIST, Daejeon, 305-701, Korea and the Cyber Convergence Security Research Division, ETRI, Daejeon, 305-700, Korea (email: youskang@etri.re.kr).

[*]J. Wang is Associate Director of the Auto-ID Lab, Fudan University, Shanghai, China (email: junyuwang@fudan.edu.cn).

[1]The first author, Dr. Engels, led the development of the Gen2 protocol under the MIT Auto-ID Center and as founding co-chair (with Dr. Chris Diorio) of the Hardware Action Group under EPCglobal.

978-1-4673-5750-0/13/$31.00 ©2013 IEEE

JTC 1/SC 31/WG 7) to develop a security framework for the Gen2 protocol. This framework admits the use of a large number of different cryptographic functions while allowing for a range of basic authentication, cryptographic access control, and secured communications functionality. The framework is in the final stages of standardization within EPCglobal while a number of cryptographic suites that provide cryptographic security by using the framework are in the early stages of standardization within ISO/IEC JTC 1/SC 31/WG 7.

Our contribution is an evaluation of the Gen2 protocol and the new Gen2 security framework against timing-based attacks such as relay attacks and Man-in-the-Middle (MITM) attacks by performing a detailed analysis of the communication timings. We determine the timing constraints on the primary commands and basic communication sequences of the Gen2 protocol and the new security framework commands, and we evaluate their ability to mitigate timing-based attacks. Theoretically, the strict timing constraints of the Gen2 protocol mitigate all relay and MITM attacks except those by attackers utilizing simultaneous, i.e., very short delay, symbol send and symbol receive capabilities. However, we found that commercial interrogators and tags do not necessarily enforce the specified timing constraints, making it possible for real-world relay and MITM attacks.

We outline two cryptographic suites that can be created for use with the new framework. We evaluate potential vulnerabilities in the protocols being used for authentication and secure communications. We conclude that the new Gen2 security framework allows for secure authentication and communication protocols; however, cryptographic suites that use the new delayed response functionality are vulnerable to relay and MITM attacks.

The remainder of this paper is organized as follows. In Section II we review the current Gen2 protocol and evaluate the timing constraints within the protocol. We then introduce the new Gen2 security framework in Section III. In Section IV we demonstrate the use of the Gen2 security framework with a cryptographic suite using AES in CBC mode. The AES OFB cryptographic suite is presented in Section V. Section VI presents the security evaluation, and we draw the relevant conclusions in Section VII.

## II. Gen2 Protocol Overview

The Gen2 protocol [5] defines an air interface between a tag and an interrogator operating within the frequency range of 860 MHz to 960 MHz. The physical layer (PHY) and the medium access control layer (MAC) are defined within the protocol for both the interrogator-to-tag communications (the downlink) and the tag-to-interrogator communications (the uplink). Additionally, the Gen2 protocol defines a set of procedures including a Reader-Talks-First (RTF) framed slotted aloha anti-collision algorithm. In this section, we review the Gen2 protocol and determine the theoretical maximum and minimum timings of several common commands. We also review the basic procedure required to identify and to singulate a tag. The timings required to communicate are evaluated and experimental results are presented. Finally, we evaluate these time windows for vulnerabilities to timing-based attacks.

### A. Gen2 Commands and Operational States

The Gen2 protocol defines a set of standard commands and communication procedures using these commands. The Gen2 protocol specifies a total of 15 commands divided into three rough categories: singulation, read/write, and security. The singulation commands include `Query`, `Select`, `QueryRep`, `QueryAdjust`, `ACK`, and `NACK`. The read/write commands include `Req_RN`, `Read`, `Write`, `Lock`, `BlockWrite`, `BlockErase`, and `BlockPermalock`. The security commands include `Kill` and `Access`.

The functionality of each tag is governed by a finite state machine defined in Figure 6.21 of [5]. The tag changes its state only in response to valid commands. The interrogator issues commands to move a population of tags through a sequence of their operational states in order to achieve some functionality such as identifying all tags. All functional sequences begin with the singulation of a tag which is initiated by the `Query` command.

After completing a power up initialization sequence, the tag enters the `Ready` state. The basic progression of the tag states during the singulation process involves the interrogator issuing a set of commands that move the tag from `Ready` to `Arbitrate` to `Reply` to `Acknowledged`. When a tag enters the `Acknowledged` state, it is identified and has been singulated. After being singulated, the interrogator may move this tag into its `Open` state and then the `Secured` state using a password procedure. Once in the `Secured` state, the interrogator may perform a number of operations with the tag.

### B. Gen2 Communication Timings

The Gen2 protocol defines the complete timing characteristics for the uplink and downlink communications. These timing characteristics include not just the data rate but also the timing between commands and their responses (T1) and the timing between responses and the next command (T2).

The interrogator may communicate to tags at a data rate as slow as 26.7 kbps (kilo-bits per second) and as fast as 128 kbps. Tag communication rates range from 5 kbps to 640 kbps with rates between 40 kbps and 120 kbps being commonly used.

It is important to note that a Preamble is issued before the `Query` command. The Preamble specifies the communication rates and requires between $51.625\mu$s and $337.500\mu$s to communicate. A Framesync is sent before all other commands. The Framesync specifies the interrogator communication rate only and requires between $34.375\mu$s and $112.500\mu$s to communicate.

Table I lists some of the most commonly used commands and their expected communication times for the fastest and slowest data rates. The `Ack` command, for example, consists of 18 bits plus a Framesync that require between $175.000\mu$s and $786.650\mu$s to communicate. The short reply to the `Ack` command consists of the *TR preamble, PC, EPC,* and *CRC16* (that is, 6 bits + 16 bits + 128 bits + 16 bits = 166 bits). At the maximum tag to interrogator bit rate of 640 kbps, it takes $259.375\mu$s for the tag to completely send 166 bits.

After receiving a command, tags are required to respond within a fixed period of time referred to as T1. T1 is dependent upon the communication rate used by the tag and has values ranging from $15.625\mu$s to $250.000\mu$s. The interrogator may assume no response is coming if it is not received within T1.

We evaluated the T1 timings of five commercial Gen2 tags using a tag verification system that utilizes a spectrum analyzer, a vector signal analyzer, and custom software to emulate both a tag and an interrogator. Table II presents the measured T1 timings from the commercial Gen2 tags using this system. All of the measurements used a Tari value of $6.25\mu$s and a DR value of 64/3. This yielded a fast T1 time that is close to the expected T1 values. For the BLF (Bit Length Frequency) of 640k, the expected T1 time is $18.75\mu$s. For the BLF of 426k, the expected T1 time is $23.47\mu$s. Ayer and Engels [10] experimentally found similar results. Additionally, they found that tags respond correctly for Tari values less than $6.25\mu$s.

Using this same system as a tag emulator, we evaluated the Impinj R2000 interrogator module and found it to be able to receive tag communications with a T1 value as small as 1ns and as large as 4.29 times the expected T1 value, although this multiple decreases with the slowest communication rates.

### C. Gen2 Command Sequence Timings

The possible data rates used for communication define the maximum and minimum timings, i.e., the time window, during which successful active attacks must occur. T1 and T2 form timing constraints respectively on the command responses and the issuing of the next command in a sequence. Different data rates have different T1 and T2 values. We investigate these time windows from the perspective of the number of bits that may be communicated during these windows and relate these to a simplified description of the singulation process.

The T1 window is as large as $250\mu$s. A command containing 28 total bits and using a Framesync requires at least $253.125\mu$s to transmit at the highest interrogator data rate. Thus, commands containing more than 27 bits take more than the maximum T1 time to communicate. Only the `Ack` and `QueryRep` commands contain less than 28 bits within them. But, the response from the tag to the `Ack` command requires at least $259\mu$s to communicate; therefore, the fastest `Ack` command and response time is $450.000\mu$s including T1 but excluding T2.

TABLE I
COMMUNICATION TIMES FOR COMMANDS IN THE GEN2 PROTOCOL.

| Command | Command ($\mu$s) 26.7 kbps | Command ($\mu$s) 128 kbps | Response ($\mu$s) 40 kbps | Response ($\mu$s) 640 kbps |
|---|---|---|---|---|
| `Select` (Framesync + 44 bits) | 1760.500 | 378.125 | No Response | |
| `Query` (Preamble + 22 bits) | 1161.47 | 223.438 | 850.000 | 34.375 |
| `Ack` (Framesync + 18 bits) | 786.650 | 175.000 | 4450.000 | 259.375 |
| `QueryRep` (Framesync + 4 bits) | 262.300 | 65.625 | 850.000 | 34.375 |
| `Req_RN` (Framesync + 40 bits) | 1610.500 | 346.875 | 1265.625 | 85.430 |

TABLE II
MEASURED T1 TIMINGS FOR FIVE COMMERCIAL GEN2 TAGS.

| Tag | BLF | Coding | T1 ($\mu$s) |
|---|---|---|---|
| Alien 9654 | 640k | FM0 | 20.48 |
| | 426k | FM0 | 25.62 |
| Avery Dennison 843 | 640k | FM0 | 17.14 |
| | 426k | FM0 | 21.66 |
| Impinj E11 | 640k | FM0 | 19.88 |
| | 426k | FM0 | 23.24 |
| Quanray 622G0021 | 640k | FM0 | 19.88 |
| | 426k | FM0 | 23.24 |
| Quanray 622GF848 | 640k | FM0 | 19.94 |
| | 426k | FM0 | 23.33 |

After receiving a tag's response, the interrogator may issue the next command not less than $4.688\mu$s and no more than $500.000\mu$s after receiving the tag's response. This is the interrogator's response time, T2. A command containing 60 total bits that is sent at the highest interrogator data rate requires at least $503.125\mu$s to transmit (including the Framesync).

Using our experimental setup and five test tags, we experimentally determined the minimum and maximum T2 times for each of the tags using a BLF of 40kbps and using a BLF of 640kbps. We determined that each of the tags would always respond to the next command if the next command was issued between $18\mu$s and $30\mu$s after the completion of the tag's response. Beyond this range the tag's response became less reliable. This result suggests a narrow time window within which highly reliable communication sequences must perform.

The stateful nature of the Gen2 tags requires that sequences of commands must be issued to tags in order to first singulate the tag and then to perform singulated operations on the tag. Our T1 results show that interrogators are very flexible on the T1 delay that they tolerate while still receiving communications from tags. However, our T2 results suggest that the tags are less tolerant of both early and delayed commands from the interrogators.

The Gen2 protocol defines several command sequences including for singulation, for writing to user memory, and for killing a tag. The singulation process is fundamental to all tag communications since it is used to identify tags and to place them within a state, either `Open` or `Secured`, that allows for additional on-tag functionality to be exercised. A simplified description of the first five steps in the singulation process is as follows [tag state information is denoted in brackets]:

1) The interrogator $\mathcal{R}$ optionally issues a `Select` command. [Tag starts and ends in `Ready`.]
2) $\mathcal{R}$ initiates a singulation procedure by sending a `Query` command.
3) A tag $\mathcal{T}$ generates a random number. If it is zero, $\mathcal{T}$ backscatters a 16-bit random number, RN16. [`Reply`]
4) Upon receiving the RN16, $\mathcal{R}$ issues an `Ack` command that includes the received RN16 value.
5) Upon receiving the `Ack` command with a matching RN16, $\mathcal{T}$ communicates its protocol control (PC), extended protocol control (XPC) (if applicable), EPC, and cyclic redundancy check (CRC). [`Acknowledged`]

At this point, the tag $\mathcal{T}$ is singulated, and it has identified itself to the interrogator by its EPC. Since the tag $\mathcal{T}$ is singulated, the interrogator may issue singulated command sequences to which only tag $\mathcal{T}$ will respond. To exercise the

new Gen2 security functionality, these singulated command sequences must begin with the `Req_RN` command being sent as Step 6. The new Gen2 security framework is described in the following section.

## III. Gen2 Security Framework

In this section, we introduce the key air interface features of the new Gen2 security framework in much the same way that we discuss the basics of the Gen2 protocol in Section II. We focus upon the commands and the timings of those commands. In the following sections we present overviews of cryptographic suites that utilize this framework to achieve a set of cryptographically secure functionality. The proposed framework does not provide for all possible security mechanisms. Instead, if focuses upon identity authentication, which allows for authenticated access to tag functionality, and secure and authenticated communication channels, which provide for confidentiality and integrity protection of commands such as `Read` and `Write`. The new Gen2 security framework commands are used according to the cryptographic suites that are being defined in ISO. At the time of this writing, this framework is defined in all but a few of its details, but it is not yet an EPCglobal standard.

The new Gen2 security framework has six primary commands that may be used with a cryptographic suite: `Challenge`, `Authenticate`, `ReadBuffer`, `SecureComm`, `AuthComm`, and `KeyUpdate`. The primary goal of these commands is to provide a framework for cryptographic identity authentication (`Challenge` and `Authenticate`) and a secured and/or authenticated communication channel (`SecureComm` and `AuthComm`). The `KeyUpdate` command allows for the explicit changing of cryptographic keys.

The new security framework allows for a tag to take an extended period of time (several tens of milliseconds or longer) to complete its operations. We refer to a response that is allowed to take more than T1 time as a *delayed response.* A tag requiring a significant amount of time to complete its operations will beacon a 'busy signal' at least every 20ms to indicate to the interrogator that it is still computing its result. A tag instead of communicating its response directly to the interrogator may write its result to a new response buffer and, after doing so, indicate to the interrogator that it is finished. The interrogator can read the contents of this buffer through the `ReadBuffer` command.

The `Challenge` command is a broadcast command that contains at least 48 bits that form a packet wrapper around a message that is to be interpreted according to some cryptographic suite. The contents of the message is defined by the cryptographic suite being used. The `Challenge` command contains fields within it that enables the interrogator to specify which cryptographic suite is to be used.

The `Authenticate` command is a singulated command that contains at least 64 bits that form a packet wrapper around a message that is to be interpreted according to some cryptographic suite. The contents of the message is defined by the cryptographic suite being used. The `Authenticate` command contains fields within it that enables the interrogator to specify which cryptographic suite is to be used.

TABLE III
Minimum communication times for selected commands in the proposed Gen2 security framework.

| Command | Time for Command ($\mu$s) 26.7 kbps | 128 kbps |
|---|---|---|
| `Challenge` (RT Framesync + 48 bits) | 1910.320 | 409.375 |
| `Authenticate` (RT Framesync + 64 bits) | 2509.600 | 534.375 |
| `SecureComm` (RT Framesync + 56 bits) | 2209.960 | 471.875 |
| `AuthComm` (RT Framesync + 42 bits) | 1685.41 | 362.500 |
| `ReadBuffer` (RT Framesync + 67 bits) | 2621.965 | 557.813 |

The `SecureComm` command is a singulated command that contains at least 56 bits that form a packet wrapper around a secure message that is being communicated to the tag. This message will typically be a command, such as a `Write` command or a `Read` command, that is encrypted and authenticated according to the cryptographic suite that was established during the authentication process.

Similarly, the `AuthComm` command is a singulated command that contains at least 42 bits that form a packet wrapper around an authenticated message that is being communicated to the tag. This message will typically be a command, such as a `Write` command or a `Read` command, that is authenticated according to the cryptographic suite that was established during the authentication process.

The `ReadBuffer` command is a singulated command that contains 67 bits and is used to retrieve the contents of the response buffer within the tag.

Table III shows the minimum communication timings of these five commands under the fastest and slowest interrogator communication rates.

## IV. The AES CBC Cryptographic Suite

In this section, we outline a simple cryptographic suite based upon the Advanced Encryption Standard (AES) operating in the Cypher Block Chaining (CBC) mode of operation. AES is a symmetric key block cipher that has been standardized by the National Institute of Standards and Technology [11]. Block ciphers require a mode of operation in order to remain secure during their use. One of the simplest modes of operation is the CBC mode of operation [12].

In the CBC mode, an initialization vector (IV) is used in the initial step of the encryption and decryption processes. The IV is XORed with the first block of plaintext $PT_1$ to be encrypted with the result of the XOR operation being encrypted by AES. The resulting block of cipher text $CT_1$ is XORed with the second block of plaintext $PT_2$ and the result of the XOR operation is encrypted and so on until all plaintext blocks have been encrypted. In AES, a block is 128 bits in length; therefore, the IV must be 128 bits in length. The IV need not be secret; however, the IV should be unpredictable [12].

Efficient implementations of the AES encryption functionality have been widely published [13][14]. The low power

achieved in these implementations makes AES encryption suitable for use on passive UHF tags; however, the large number of clock cycles required to achieve this low power, 160 clocks in [14], does not allow a power efficient AES implementation to meet T1 timing requirements for all communication speeds. A typical passive UHF RFID tag has a 2MHz clock. Therefore, only the encryption of one block will require approximately $80\mu$s, or more than four times the minimum T1, to complete.

A full description of the proposed AES-CBC cryptographic suite is beyond the scope of this paper. We consider in this paper a simplified mutual authentication process only. In the mutual authentication process we require that both the interrogator and the tag contribute data in order that they may authenticate each other while providing for mitigation against certain attacks. We require that the tag use only the AES encrypt function and the CBC mode of operation. And, for simplicity and speed of communications, we will allow ourselves to have a less than 128-bit level of security.

The mutual authentication process begins after singulation has occurred, so we begin with Step 6 of the communication process that we presented in Section II.

6) The interrogator issues the `Req_RN` command Note, for simplicity we assume that the interrogator obtains the shared secret key for the identified tag at this point.
7) The tag responds with its Handle. [Open]
8) The interrogator issues the `Authenticate` command with the value in the message being a 64-bit random number $RN64_{\mathcal{R}}$ and an indication that this is the first mutual authentication step.
9) The tag sets its IV equal to $RN64'_{\mathcal{R}}$, the value the tag received in the message field of the `Authenticate` command. The tag generates its own 64-bit random number, $RN64_{\mathcal{T}}$, and encrypts it to get $CT_1$. The tag sends to the interrogator $RN64_{\mathcal{T}}$ and $CT_1$. [Open]
10) The received $CT'_1$ is decrypted to obtain $RN64'_{\mathcal{T}}$. If $RN64'_{\mathcal{T}}$ is equal to the received 64-bit value, then the tag has authenticated itself to the interrogator. The interrogator then authenticates itself to the tag by encrypting 0 to get $CT_2$, and issuing a second `Authenticate` command with the value in the message being the 64 least significant bits of $CT_2$ where the message contains an indication that this is the second mutual authentication step.
11) The tag encrypts 0 to get $CT'_2$. If the 64 least significant bits of $CT'_2$ are equal to the 64 bits received in the second `Authenticate` command, then the interrogator has authenticated itself to the tag, and the tag responds with the 64 most significant bits of $CT'_2$. [Secured]
12) If the 64 most significant bits of $CT_2$ are equal to the 64 bits received from the tag, then the tag is authenticated.

This is a simple mutual authentication process that requires two `Authenticate` commands. With the first `Authenticate` command, the interrogator sends at least 64 bits of message contents in addition to the 64 bits required of the `Authenticate` command itself. The tag's response contains at least 192 bits of data (the 64-bit $RN64_{\mathcal{T}}$ and 128-bit $CT_1$). At the conclusion of the first `Authenticate` command, the tag has authenticated itself to the interrogator by using values generated by both the tag and the interrogator.

The second `Authenticate` command is used to authenticate the interrogator to the tag. Provided that the XOR of the random number tuple $< RN64_{\mathcal{R}}, RN64_{\mathcal{T}} >$ is unique, the cipher text generated by encrypting 0 is not predictable. The tag's positive response, the 64 most significant bits of $CT_2$, serves to provide a positive acknowledgement to the interrogator and to provide for a continuous authentication of the tag within the communication sequence.

The shared secret key is the only secret in this authentication process. We chose to limit the total number of bits sent in any one communication while providing a reasonable level of security. A higher level of authentication security may be obtained by using at least 80-bit random numbers and by issuing the second `Authenticate` command two or more times in sequence. This may be defined as a high security version for this cryptographic suite.

The defined authentication process requires the tag to perform one encryption after receiving a valid `Authenticate` command. Since this computation may take more than time T1 at the fastest communication rates, for this cryptographic suite we must allow for a delayed response to the `Authenticate` command.

## V. The AES OFB Cryptographic Suite

The Output Feedback (OFB) mode of operation [12] does not utilize AES to operate directly on the plaintext to be encrypted. Instead, in OFB mode, the IV is encrypted by the block cipher in order to generate a key stream, e.g., the output from encrypting the IV with AES is a 128-bit key $O_1$. The key stream $O_1$ is XORed with the plaintext $PT_1$ in order to generate the cipher text $CT_1 = PT_1 \oplus O_1$. Subsequent bits of the key stream are created by encrypting the prior key stream blocks, e.g., $O_2$ is created by encrypting $O_1$ with the AES block cipher.

Unique IV's must be used with AES-OFB [12]. Repeating an IV allows confidentiality and integrity attacks to be successful.

A full description of the AES-OFB cryptographic suite is beyond the scope of this paper. We consider in this paper a simplified mutual authentication process only. In the mutual authentication process we require that both the interrogator and the tag demonstrate knowledge of a key stream, $O_1$, that is generated based upon a 64-bit random number, $RN64_{\mathcal{T}}$, created solely by the tag.

While it is desirable to have $O_1$ dependent upon values generated by both the interrogator and the tag, having it dependent upon the tag's random number only allows $O_1$ to be generated during the tag's Power Up initialization process. This allows the tag to not execute any AES operations during at least the first step of the mutual authentication process.

The mutual authentication process begins after singulation has occurred. We assume that $RN64_{\mathcal{T}}$ and $O_1$ are created during the tag power up.

6) The interrogator issues the `Req_RN` command Note, for simplicity we assume that the interrogator obtains the shared secret key for the identified tag at this point.
7) The tag responds with its Handle. [Open]
8) The interrogator issues the `Authenticate` command with the message being a 64-bit random number $RN64_{\mathcal{R}}$ and an indication that this is the first mutual authentication step.
9) The tag generates $CT_1 = O_1^{\text{bits 128-65}} \oplus RN64'_{\mathcal{R}}$. The tag sends to the interrogator $RN64_{\mathcal{T}}$ and $CT_1$. [Open]
10) The interrogator generates its $O'_1$ by encrypting $RN64'_{\mathcal{T}}$. It then decrypts $CT_1$ and compares the result with $RN64_{\mathcal{R}}$. If

they match, then the tag has authenticated itself to the interrogator. The interrogator issues an `Authenticate` command with the value of the message being the next 64 bits in the key stream, 64 least significant bits of $O_1'$, and where the message contains an indication that this is the second mutual authentication step.

11) The tag compares the received 64 bits of the message with the 64 least significant bits of $O_1$. If these values are equal, then the interrogator has authenticated itself to the tag. The tag responds with $O_2$. [Secured]

This simple mutual authentication process relies solely upon the tag to generate the IV. Therefore, the AES-OFB cryptographic suite does not require a delayed response during the response to the first `Authenticate` command. A delayed response may be used in the response to the second `Authenticate` command, but both tag and interrogator are authenticated at this point. Furthermore, $O_2$ may be computed during power up.

A higher level of authentication security may be obtained by using at least 80-bit random numbers and at least 80 bits of key stream in the second `Authenticate` command. This may be defined as a high security version for this cryptographic suite.

Additionally, the `Challenge` command can be used to provide sufficient time for key stream generation. For the AES-OFB cryptographic suite, we define a message payload for the `Challenge` command that indicates to the tag to generate the $O_1$ and $O_2$ key stream with the IV being the concatenation of its $RN64_{\mathcal{T}}$ plus the $RN64_{\mathcal{R}}$ contained within the `Challenge` message. This `Challenge` command would be sent as Step 0 in the communication sequence. All other steps in the mutual authentication communication sequence would remain unchanged.

Clearly, if the tag repeats its $RN64_{\mathcal{T}}$ then it will repeat the key stream used during the initialization process. Therefore, $RN64_{\mathcal{T}}$ should either be chosen uniformly at random in a cryptographically secure manner, or an 80-bit or 128-bit random number should be used.

## VI. Security Evaluation

We briefly consider only four of the possible attacks on a Gen2 system: eavesdropping, snooping, relay, and MITM attacks. These attacks help to clarify the limits of the proposed Gen2 security framework. We focus upon the timing issues related to the Gen2 communications. The tight timing constraints of the Gen2 protocol combined with the large message sizes of the security commands mitigates capture and forward relay and MITM attacks even for the loose timings allowed by commercial interrogators. More sophisticated attacks that both capture and send simultaneously may be successful against even the basic Gen2 systems. The use of the delayed response by a cryptographic suite makes that suite vulnerable to relay and MITM attacks.

### A. Eavesdropping Attacks

Eavesdropping attacks are one of the simplest types of passive attacks. In an eavesdropping attack, an attacker passively listens and records the communications that transpire between interrogator and tag on both the downlink and the uplink. These captured communications can be used for more complex attacks such as replay attacks and track and trace attacks.

The new Gen2 security framework as we have presented it here is not designed to mitigate eavesdropping attacks during the singulation process. The framework operates primarily after singulation has occurred. Custom commands and functionality that provide for eavesdropping mitigation during singulation are beyond the scope of this paper.

The new `SecureComm` command is designed to mitigate eavesdropping attacks that occur after a secure communication has been established. `SecureComm` provides for an encrypted and authenticated communication channel, both downlink and uplink, that protects the confidentiality and the integrity of the communications.

Eavesdropping attacks should be assumed to happen during the operation of any Gen2 system. For the cryptographic suite designers, this awareness requires that cryptographically secure functionality should not be easily captured by an attacker and replayed to an unsuspecting victim. In the context of the mutual authentication processes presented for the AES-CBC and AES-OFB cryptographic suites it is clear that a mitigation mechanism to eavesdropping and replay attacks is the use of values from both the interrogator and the tag during the mutual authentication process.

If a single mutual authentication process is to be replayed, either emulating the tag or emulating the interrogator, there is a 1 in $2^{64}$ chance that the victim device will choose the captured $RN64$. This probability decreases with larger bit random numbers. With the slow nature of even the fast communication links in the Gen2 protocol, it is unlikely that either a tag or an interrogator will choose the same $RN64$ within a reasonable lifetime of the system. Therefore, eavesdropping attacks, while useful as the basis for track and trace attacks, are unlikely to be successful in attempts to capture a sufficient number of responses to mount successful replay attacks for authentication purposes.

### B. Snooping Attacks

Snooping attacks are a simple active attack where an attacker attempts to communicate with a tag. The Gen2 protocol is designed to have promiscuous tags; therefore, any Gen2 interrogator can be used for snooping attacks.

All Gen2 tags respond to valid commands and will go through the singulation and identification process for any interrogator. The new Gen2 security framework as we've discussed it here does not prevent a snooping attacker from performing the singulation process to retrieve a tag's EPC. The framework is designed to mitigate any access to tag functionality beyond the tag's EPC. Therefore, the authentication portion of a cryptographic suite should be resistant to repeated snooping attacks.

In a snooping attack aimed at gaining access to a tag's user memory and other functionality, the attacker will control the interrogator's portion of the communication. By holding values such as the $RN64_{\mathcal{R}}$ constant, the snooping attacker may capture a large number of tag responses corresponding to this constant.

The singulation process takes at least 540$\mu$s for a single tag (Steps 1 through 5), and the mutual authentication process takes significantly longer to complete. The `Authenticate` command with a 64 bit message takes at least 1034$\mu$s to send at the fastest data rates. The tag's response to the first `Authenticate` command in the AES-CBC protocol requires at least 336$\mu$s to communicate. For the AES-OFB protocol, this response requires at least 243$\mu$s. The complete sequence of two `Authenticate` commands requires at least 2555$\mu$s for the AES-CBC cryptographic suite and at least 2367$\mu$s for the AES-OFB cryptographic suite.

With these communication times, it will take a snooping attacker more than 6 million years just to issue $2^{64}$ first `Authenticate` commands (over 7 million years if the attacker waits for the response). It will take more than 1.7 years for the attacker to obtain even a reasonable number of responses, which we estimate at $2^{32}$ responses. Remember that these timings are just for a single value of $RN64_{\mathcal{R}}$ and only capture the tag's response to the first `Authenticate` command.

If only tag authentication is desired, then by varying $RN64_{\mathcal{R}}$ a result for each sent number will be obtained. Given the extreme timings, it is to the luck of the attacker if either the $RN64_{\mathcal{R}}$ chosen by the interrogator can be guessed or is chosen such that it corresponds to one of the results previously snooped by the attacker. In order to minimize the chance of an attacker guessing the $RN64_{\mathcal{R}}$ chosen by the interrogator, we recommend that the interrogator generate a random number chosen uniformly at random. We discourage the use of pseudorandom number generators, counters, and time as the value chosen by the interrogator for $RN64_{\mathcal{R}}$.

### C. Relay Attacks

A relay attack involves an attacker relaying messages between an interrogator and a tag; thereby allowing the attacker to impersonate the identity, credentials, and functionality of the interrogator and the tag that are actually communicating. A relay attack bypasses any application layer security including all security that is enabled by the new Gen2 security framework. Even cryptographic suites that are based on strong cryptographic principles may be defeated by a relay attack. For example, an attacker can circumvent the authentication protocols presented for AES-CBC and AES-OFB simply by relaying all communications between a tag and an interrogator, assuming that all communication timing constraints are met. Thus, the attacker may not know the contents of the communications, but the attacker is able to reap some benefits of the communications such as unlocking of a door. Consequently, the security enabled by the new Gen2 security framework will be bypassed with a successful relay attack.

Hancke and Kuhn [15] correctly identify that tight timing constraints on the communications between an interrogator and a tag provide an effective countermeasure to relay attacks. In the Gen2 protocol, the T1 and T2 timing constraints create tight timing constraints that act as countermeasures to relay attacks.

During the singulation and identification process, the tag's 166 bit response in Step 5 requires more than 250$\mu$s to communicate to the interrogator. When combined with the timing of the `Ack` command and the minimum T1 and minimum T2, the interrogator can theoretically detect a timing violation that would result from any basic relay of this command and response. The only way an attacker cannot violate the T1 timing constraint is to instantly send and receive every bit without any delay.

In order to overcome this T1 error inherent in the singulation process, an attacker can singulate a victim tag and simply replay the singulation conversation to the victim interrogator. However, this approach cannot be performed for the authentication process.

The `Authenticate` command contains at least 64 bits plus a Framesync that require at least 534$\mu$s to communicate. This is more than twice the maximum T1. Basic authentication mechanisms such as those proposed for AES-CBC and AES-OFB require at least 64 bits of data from the interrogator. With at least 128 bits plus a Framesync to be sent, the `Authenticate` command takes at least 4.28 times the maximum T1 time. Even with the relaxed T1 timings implemented within commercial interrogators, the relay of this most basic of `Authenticate` commands will result in a T1 violation at the victim interrogator for both of the presented cryptographic suites.

The presented `Authenticate` commands require at least 128 bits to be sent requiring more than $1000\mu$s to communicate from the interrogator to the tag. A tag becomes less reliable in its communications the longer it is held with power but without communications. While a tag may respond after a significant T2 violation, the reliability of this communication decreases significantly, and waiting for more than $1000\mu$s will allow tags enough time to detect a T2 violation while minimizing their ability to detect and respond to the command.

Relay attacks are mitigated by even weak `Authenticate` commands that do not utilize a delayed response. Therefore, the T1and T2 timing constraints of the Gen2 protocol act as effective countermeasures to relay attacks. The presented AES-CBC and AES-OFB cryptographic suites provide for significant timing delays in both the downlink and the uplink due to the large number of bits that are communicated during authentication. Furthermore, interrogators that communicate at or near the fastest communication rates will further mitigate relay attacks.

However, the use of a delayed response, such as may be used during the AES-CBC mutual authentication process, provides the opportunity for relay attacks to be successful; therefore, the delayed response should not be used.

### D. Man-in-the-Middle Attacks

Related to the relay attack is the Man-in-the-Middle attack. MITM attacks are a type of attack where attackers intrude into a communication connection to intercept the communications and possibly inject data into the communications. An MITM attack involves the attacker making independent connections with the interrogator and the tag and relaying messages between them (just like a relay attack). Unlike a relay attack, in the MITM attack, the attacker may access and modify any and all data that is communicated.

MITM attacks vary in when data is modified and what data is modified. The deciding factors are the communication protocol that is being used and the degree to which an MITM attacker wishes to inject data into the communications. Song et. al. [16] propose an MITM attack on the security protocol based upon AES using OFB described in [17]. The proposed attack relies upon the relay of a custom command `Sec_ReqRN` which has 35 bits within the command. As we have seen, the relay of a command with 35 bits exceeds the maximum available T1 time; therefore, the MITM would be detectable by an interrogator monitoring for T1 violations. Furthermore, the relay of the `Authenticate` command with or without modifications will cause T1 and T2 timing violations even for very relaxed interrogator timings of commercial interrogators.

MITM attacks on Gen2 systems, therefore, must be designed so as to minimize the number of commands and responses that must be relayed, or the relays must be performed during commands utilizing a delayed response. The use of the delayed response by a victim interrogator in communicating using the AES-CBC mutual authentication protocol would allow an MITM attack to operate within this process. The AES-OFB mutual authentication protocol, in contrast, does not require the use of a delayed response, and any command relays in an MITM attack would cause T1 and T2 timing violations.

MITM attacks on the `SecureComm` and the `AuthComm` commands are mitigated by immediate responses that can be provided by AES with OFB mode. Relaying either of these commands will cause a T1 and a T2 violation. However, using a delayed response, such as might be used with the AES-CBC cryptographic suite, will allow MITM attacks. We note that AES-OFB would benefit from a custom NOP (no operation) command that is designed simply to allow the tag to generate more of the key stream.

## VII. Conclusions

We have presented a summary of the Gen2 protocol and the new Gen2 security framework. And, we have presented simple cryptographic suites and evaluated their theoretical and actual timings in an evaluation of possible attacks on the Gen2 system. We have evaluated the possibility of relay attacks and MITM attacks on RFID systems operating according to the Gen2 protocol and the new framework. The Gen2 protocol utilizes tight T1 and T2 timing constraints during its communication process. These constraints act as an effective countermeasure to relay attacks and MITM attacks that require commands or their responses to be relayed. The delayed response contained within the new Gen2 security framework allows for slow cryptographic operations, but it creates a new vulnerability to relay attacks and MITM attacks for cryptographic suites that utilize a delayed response.

Security functionality built on top of the new Gen2 security framework that adheres to the tight T1 and T2 timing constraints will maintain the Gen2 countermeasures to relay attacks and MITM attacks. For this reason, we strongly encourage that any security functionality that is developed for use with the Gen2 protocol maintain the T1 and T2 timing constraints in all, or at least a crucial few, security commands. Failure to adhere to the T1 and T2 timing restrictions or using excessively short commands will make this ill-designed security functionality vulnerable to attacks that exploit loose timing constraints. We further recommend that interrogators communicate as fast as possible without using delayed responses to minimize the time available for relay attacks and MITM attacks.

## Acknowledgements

The authors thank Hwa-Tech Information System Co., Ltd. for providing test environment and technical support for the experiments, and Lingzhi Fu and Min Li from the Auto-ID Lab at Fudan University.

## References

[1] K. Romanow and S. Lundstrom. RFID in 2005: The what is more important than the Wal-Mart edict. AMR Research Alert, AMR Research, 2003.

[2] D.W. Engels, J. Foley, J. Waldrop, S.E. Sarma, and D. Brock. The networked physical world: An automated identification architecture. In *Proceedings of the Second IEEE Workshop on Internet Applications (WIAPP 2001)*, pages 76–77, 2001.

[3] Sanjay E. Sarma, Stephen A. Weis, and Daniel W. Engels. Low cost RFID and the electronic product code. In *CHES '02, Workshop on Cryptographic Hardware and Embedded Systems, LNCS*. Springer, 2002.

[4] EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz – 960MHz, Version 1.0.9. EPCglobal Inc., January 2005. www.gs1.org.

[5] EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz – 960MHz, Version 1.2.0. EPCglobal Inc., October 2008. www.gs1.org.

[6] ISO/IEC 18000-6:2010, Information technology – Radio frequency identification for item management–Part 6: Parameters for air interface communications at 860MHz to 960MHz. International Organization for Standardization, April 2011. http://www.iso.org.

[7] Daniel Bailey and Ari Juels. Shoehorning security into the EPC tag standard. *Security and Cryptography for Networks*, pages 303–320, 2006.

[8] Ari Juels. RFID security and privacy: A research survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, February 2006.

[9] K. Koscher, A. Juels, V. Brajkovic, and T. Kohno. EPC RFID tag security weaknesses and defenses: passport cards, enhanced drivers licenses, and beyond. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 33–42. ACM, 2009.

[10] Nikhil Ayer and Daniel W. Engels. Evaluation of ISO 18000-6C Artifacts. In *Proceedings of IEEE International Conference on RFID*, pages 123–130, April 2009.

[11] National Institute of Standards and Technology (NIST). Advanced encryption standard (AES). Federal Information Processing Standards Publication 197, November 26, 2001.

[12] Morris Dworkin. Recommendation for block cipher modes of operation: Methods and techniques. NIST Special Publication 800-38A, December 2001.

[13] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES implementation on a grain of sand. *IEE Proc. Inf. Sec*, 153(1):13–20, October 2005.

[14] P. Hämäläinen, T. Alho, M. Hännikäinen, and T. Hämäläinen. Design and implementation of low-area and low-power AES encryption hardware core. In *Ninth Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD 2006)*, pages 577–583. IEEE Computer Society, 2006.

[15] G.P. Hancke and M.G. Kuhn. An RFID distance bounding protocol. In *Proceedings of IEEE/CreateNet SecureComm*, pages 67–73, September 2005.

[16] Boyeon Song, Jung Yeon Hwang, and Kyung-Ah Shim. Security Improvement of an RFID Security Protocol of ISO/IEC WD 29167-6. *IEEE Communications Letters*, 15(12), December 2011.

[17] ISO/IEC WD 29167-6, Information technology – Automatic identification and data capture techniques – Part 6: Air Interface for security services and file management for RFID at 860-960 MHz. International Organization for Standardization, May 2010. http://www.iso.org.