Shoehorning Security into the EPC Standard

Daniel V. Bailey and Ari Juels

RSA Laboratories
Bedford, MA 01730, USA
e-mail: {dbailey,ajuels}@rsasecurity.com

23 January 2006

Abstract. The EPCglobal Class-1 Generation-2 UHF tag standard is certain to become the *de facto* worldwide specification for inexpensive RFID tags. Because of its sharp focus on simple "license plate" tags, it supports only the most rudimentary of security and privacy features, and essentially none of the cryptographic techniques that underpin authentication and privacy-protection in higher-powered computational devices. To support more-sophisticated applications, the drafters of this standard envisioned the re-use of the basic air interface and command set in higher-class standards.

We propose ways to incorporate mainstream cryptographic functionality into the Class-1 Gen-2 standard. Our techniques circumvene the intended modes of operation of the standard, but adhere closely enough to preserve formal compliance. For this reason, we use the term *shoehorning* to describe our layering of new security functionality on the standard.

Key words: authentication, cloning, counterfeiting, EPC, PIN compromise, RFID

1 Introduction

Radio Frequency IDentification (RFID) tags promise in the near future to become the most numerous computational devices in the world. Their impending pervasiveness owes much to the power and flexibility that they achieve through starkly minimalist design. In their most basic form, RFID tags are little more than wireless barcodes that facilitate the tracking of objects in supply chains – at present, generally bulk containers like crates.

Many industries are embracing a recently ratified standard for RFID tags called the EPCglobal Class-1 Generation-2 UHF tag standard [12]. *EPC tags*, as the tags compliant with this standard are called, seem certain to become the *de facto* standard for low-cost RFID. It is projected that Class-1 Gen-2 EPC tags will soon cost in the neighborhood of five cents apiece, and will number in the billions. Their basic purpose is to improve supply-chain visibility, meaning that they will furnish highly accurate real-time data on the whereabouts of objects. In contrast to barcodes, which are difficult to scan without precise object positioning and thus human intermediation, RFID tags transmit data automatically and wirelessly.

It seems natural to appeal to RFID to improve infrastructural security. Indeed, the United States Food and Drug Administration is promoting the use of EPC tags to facilitate the compilation of item pedigrees in the pharmaceutical supply chain in an effort to combat counterfeit and gray-market products. It is to be expected that other industries will likewise explicitly or implicitly draw on EPC tags as a security tool.

EPC tags *per se*, however, are poorly endowed as security devices. Apart from some rudimentary protocols that reduce over-the-air information leakage, they have only two basic security features:

1. **The "kill" command:** The EPC standard envisions that tags will eventually track individual consumer items in the supply chain. In order to protect consumer privacy, the standard provides for tags to be disabled at the point of sale in retail environments. When a reader transmits a special "kill" command to an EPC tag, along with a tag-specific, 32-bit PIN, the tag self-destructs; that is, it never again responds to reader interrogation. (Dead tags, of course, offer nearly impeccable RFID privacy.)
2. **Read/write access:** An optional feature in the EPC standard provides for access-controlled memory in EPC tags. In order to read and write to certain memory locations, an EPC reader must furnish a tag-specific PIN.

These two forms of PIN-based access control reduce the risk of certain types of attack, like malicious killing of tags, and unauthorized access to the contents of tag memory. EPC tags, however, are vulnerable to a range of other, elementary attacks. EPC tags emit static, unique identifiers, as well as data like that traditionally found in a printed barcode, namely a manufacturer name and product type. Thanks to their identifiers, EPC tags are subject to clandestine tracking; with a network of readers, an entity can correlate sightings of an individual tag – and thus potentially track its bearer. The product information on tags creates a risk of surreptitious inventorying; a reader can in principle determine what items a person is carrying with her. Such risks have been a flashpoint of concern for civil libertarians.[1]

The vulnerability of RFID tags to *cloning* has received somewhat less attention. EPC tags, in particular, release their identifiers and product information – known as EPC codes – in a promiscuous manner. Any reader may scan any EPC tag; no access control exists on EPC codes. Consequently, having scanned a target EPC tag once, a reader can harvest all of the information needed to duplicate that tag in its essentials. It is unclear whether field-programmable, i.e., blank EPC tags, will be a regular commercial offering, although it is not inconceivable. An attacker could easily imprint such a tag to create a counterfeit, i.e., duplicate EPC tag.[2] Even without blank tags, however, it is an elementary matter to create wireless devices that may not have the same physical appearance as EPC tags, but perfectly simulate their output.

The drafters of the EPC standard were aware of these privacy and security concerns. They rejected potential countermeasures, like cryptographic functionality, in favor of low cost. Rather than incorporate security technologies into Class-1 tags, EPCglobal instead imagined a hierarchy of tags [12], each successive level adding functionality while incorporating all the features of lower-class tags. In this way, higher-class tags could build on the existing infrastructure without the need to develop a new air interface for each. By way of analogy, consider the long sequence of standards under the IEEE 802.11 banner. A common command set has been extended multiple times and adapted to different air interfaces, all while leveraging past investment and (when possible) maintaining backward compatibility and coexistence.

1. **Class-1: Identity Tags** Passive-backscatter tags offering only basic features like a fixed EPC identifier, a tag identifier, kill function, and optional password-protected access control
2. **Class-2: Higher-Functionality Tags** Passive tags with all of Class-1's features and extended tag identifier and user memory, as well as *authenticated access control*

[1] As noted above, EPC tags are unlikely to see widespread use on consumer products for some years. Consumers regularly carry other types of RFID tags on their persons, however, such as payment devices and proximity cards, i.e., RFID devices that unlock doors.

[2] It is even possible that the pre-programmed data in an EPC tag could be directly modified.

3. **Class-3: Semi-Passive Tags** with all of Class-2's features as well as sensors and on-tag power sources like batteries
4. **Class-4: Active Tags** with all of Class-3's features as well as tag-to-tag communications and ad-hoc networking

In contrast to established HF RFID standards like ISO 14443 and ISO 15693, where security protocols have already been deployed, the Class-1 Gen-2 UHF air interface is designed to offer longer range, better handling of dense tag and reader environments, and lower cost. These factors will draw security applications to this standard just as they have driven its success in supply chains – as well as the tremendous expected economies of scale.

Our work In this paper, we consider various ways in which it is possible to create RFID tags that perform cryptographic functionality *while remaining compliant with both the Class-1 Gen-2 standard and conformance specification [13]* and while extending the command set. Our techniques could serve as an alternative to the creation of a Class-2 EPC standard – or as the basis for such a standard.

Our key idea is to take an expansive view of EPC tag memory. Rather than treating this memory merely as a form of storage, we consider its use as an input/output medium capable of interfacing with a cryptographic module within the tag. Read and write commands to the tag, therefore, may be commandeered to carry cryptographic values, such as messages in a challenge-response protocol. We focus on protocols for tag authentication, rather than privacy-enhancing protocols.[3]

Organization

In section 2, we briefly review related work on RFID security. We outline desirable security-service enhancements to the EPC standard in section 3. We explore the scope of the Class-1 Gen-2 EPC standard in section 4, and in section 5, propose an example cryptographic command set that may be fit into the standard. We conclude in section 6 with a brief summary.

2 Related work

Privacy has been perhaps the major security focus in the RFID literature and in press coverage as well. A number of approaches have been proposed, including simple RF shielding (e.g., aluminum foil), distance detection [8], interference with RFID singulation [22] (i.e., the standard process by which readers establish one-to-one communication with tags), rotating pseudonyms [17], physical disablement [25], proxying [30, 23], trusted computing [26], and cryptographic protocols, e.g., [2, 10, 27, 28]. Cryptographic approaches to user privacy based on symmetric-key primitives tend to be unsatisfactory from a practical standpoint. They rely on readers performing intensive searches over databases of tag keys, or else sharing of secrets across tags that can weakens their security guarantees. Public-key-based protocols are expensive. (See [20] for an overview.) For these reasons, we focus here instead on the more technically tractable problem of *authentication*.

Several researchers have proposed new, lightweight cryptographic primitives aiming at RFID authentication [18, 24, 33]. A European project [6] aims to identify new stream ciphers; some of these are potentially lightweight enough for inclusion in low-cost devices. It is as yet unclear whether any

[3] As an example of a privacy-enhancing protocol consonant with the principles we enunciate here, see [17], which proposes a system of cryptographically changing EPC codes.

of these recently proposed primitives are both strong enough and agile enough for use in low-cost RFID tags, but they represent an important continuing area of inquiry. Feldhofer et al. [7] have described an AES implementation designed specifically for RFID devices. This implementation requires security resources exceeding those presently possible in EPC tags, but perhaps suitable for some of the enhancements we describe here.

Some RFID tags do employ cryptographic primitives for authentication. These tags tend to be more expensive than EPC tags, but demonstrate the need for strong authentication in niche applications. They also demonstrate that design of good cryptographic protocols for RFID requires careful attention [3, 21].

The Auto-ID Lab, the research arm of EPCglobal, operates a special interest group devoted to use of RFID to combat counterfeiting. Researchers there have proposed uses of EPC to combat counterfeiting of consumer items [31]. They suggest that track-and-trace technologies, i.e., supply-chain monitoring based on current EPC tags, can yield good improvements over existing security. They also discuss the benefits of challenge-response protocols for tag authentication, and review extensions to existing EPC architecture for this purpose. They do not investigate incorporation of cryptography into Class-1 Gen-2 EPC tags, however. They instead propose support in future EPC standards, such as the Class-2 EPC standard.

Juels proposes ways to leverage the PIN-controls for killing and read/write access to achieve *ad hoc* authentication in Class-1 Gen-2 EPC tags [19]. The resulting protocols are cryptographically weak, e.g., they are vulnerable to eavesdropping attacks, but they permit authentication of EPC tags that would otherwise not be possible. In spirit, that work is similar to our proposals here in that it aims to leverage the existing EPC standard to achieve stronger security functionality.

Of course, it is common practice to repurpose or coopt communication-protocol standards as we propose here. As seen in the past, the broad deployment of a communications standard yields many uses beyond the imagination of its original designers. Perhaps the most notable example in recent times is the TCP/IP suite of networking protocols. Originally designed for communication among mainframe and minicomputers housed in government labs, it is now supporting transmission of video clips to cell phones and replacing the traditional public switched telephone and cable-television networks.

What is unusual about our work here is the very constrained nature of the protocol set that we propose to coopt. The EPC standard specifies an *artifact*, i.e., a device with a fixed command set specified down to the bit level, and virtually no margin for extensions and no underlying intention to support them. Yet our goal is to achieve *general, extensible security services* within the EPC standard. We require a large shoehorn indeed – but thankfully one of essentially simple design.

3 Security Services

There are of course many desirable security services for RFID authentication. Roughly speaking, they fall into three categories: *device authentication*, *device-binding authentication*, and *data-origin authentication*. In the appendix, we briefly describe these different types of authentication and explain how they introduce needs for flexible cryptographic authentication services.

4 Shoehorning

The huge economies of scale will drive down the cost of tags, readers, and their components. The low cost of components will lead to their inclusion in many devices beyond the simple "license plate" item-identification application. The extension of the Class-1 Gen-2 standard to meet these needs, including anti-counterfeiting, requires a slightly different view of the specification. Instead of implying the characteristics of an artifact that implements the protocol, we can view it simply as a communication protocol. With this approach, Class-1 Gen-2 offers a logical and physical layer protocol which can be used to carry bulk data, including that of higher-layer protocols.

To implement the security services needed especially in pharmaceutical applications, we could develop new customized extensions to the logical layer, similar to the 802.11i [11] effort. But as experience has shown, this is not a trivial task. Simply taking an otherwise secure cipher and using it to encrypt data can lead to an insecure protocol [1]. Moreover, doing so presents a difficult choice: either select a single set of algorithms all implementors must use, or provide a negotiation scheme. Fortunately, several standard interfaces have been devised for secure communications with simple devices. Given their broad deployment, they have been thoroughly implemented and analyzed, and can be applied to our task at hand.

4.1 A Simple Protocol for Entity Authentication

The Class-1 Gen-2 protocol already has a limited protocol for entity authentication: in order to access protected memory or privileged commands like kill, the reader must present a static password. In principle, to authenticate itself to the reader, the tag could do the same: we could imagine a new command that would request a password from a tag. But we observe that having the tag present static data like a password provides no additional security services than providing the EPC. Presumably, an attacker who is able to clone an EPC could just clone the password as well.

With this observation in hand, we provide a motivating example of the flexibility of our approach. Challenge-response protocols prevent an eavesdropping attacker from obtaining a static password and simply reusing it. In such a protocol depicted below, the tag computes a 32- or 64-bit response $R_T = H(K_{TS}, C_R)$ where $H()$ is a cryptographic function like a block cipher, K_{TS} is some secret key known to the tag and the reader (or server), and C_R, is a unique challenge. Of course, R_T could be chosen to have a length longer than 64 bits if conditions warrant. In an application where an attacker could feasibly try such a large number of interactive queries with the reader, a longer R_T value would be a good choice, but 64 bits is appropriate for many applications given the relatively short range of Class-1 Gen-2. To address off-line attacks, one can choose K_{TS} to be much longer – such as 128 bits – without increasing the number of bits sent over the air. We have seen several implementation reports of block ciphers like AES adapted to the severe constraints of passive RFID [7] which could serve as our function $H()$.

An extraordinary number of challenge-response protocols have been developed to suit various needs and resist various attackers. This one is presented as an example because of simplicity and a particular quirk of the Class-1 Gen-2 standard: tags do not have a method to obtain the identity of a reader. For its part, the Electronic Product Code carried by the tag is denoted ID_T.

1. $Reader \rightarrow Tag : C_R$
2. $Tag \rightarrow Reader : ID_T, R_T$

There are several types of challenge-response protocols classified by how the value C_R is chosen. Perhaps the most familiar method is for the value C_R to be chosen by the reader and explicitly sent to the tag, which we'll explore in much more detail below. In a special case called a time-synchronous one-time password, however, if the tag has a real-time clock, then it can use the time of day as an implicit challenge. This approach eliminates the need for a special message from the reader carrying C_R. To ensure a password is not being replayed, one can choose a time interval for C_R short enough to preclude replay attacks and the reader can store the last correct password value received from the tag.

Given this capability, our two-message protocol above can be collapsed into a single message: when asked for its EPC in Read, ACK, or any other command, the tag responds with its EPC concatenated with its one-time password.

1. $Tag \rightarrow Reader : ID_T, R_T$

Since according to Section 6.3.2.10.2.4, the transmitted EPC data field may be up to 512 bits, using 32 or 64 of these for a one-time password still leaves a tremendous number of available identifiers. No modifications to the spec are required, save perhaps a general agreement on the placement of the one-time password R_T within a transmitted EPC field. In this way, the tag provides additional evidence of its identity which the reader may check or not. For high throughput applications, the reader can simply ignore the one-time password value, only checking the password when it wants to gain assurance that the tag has not been cloned.

If our application requires more robust reader authentication, we could additionally require the reader to respond to a challenge. The Class-1 Gen-2 standard already provides data fields for the reader to transmit 16-bit passwords. Nothing prevents us from using a one-time password instead, verifying the provided value on the tag. In practice, the tag needs either a real-time clock or a way to deliver a challenge to the reader. In addition, given the fact that different applications need different security services and/or algorithms, we need a way for the tag and reader to negotiate a common set of features as in an SSL cipher suite [5]. Fixing a single algorithm for all applications for all time seems short-sighted since we know algorithms - even those trusted by governments and large financial institutions - get broken from time to time. We can of course define frame formats for all these things, but we quickly find that we are creating a complete customized security layer, when there are robust tools already in existence that can help.

4.2 Protocol Convergence

In contrast to typical communication protocols, Class-1 Gen-2 lacks a command to simply send bulk data over the air. In fact, most data payload fields in the protocol are limited to sixteen bits in length. This design choice is guided by the challenging environment faced by tags applied to fast-moving consumer goods. Many use cases involve hundreds or thousands of tags arranged on pallets and speeding toward a dock door. The uncertainties of antenna orientation together with the sheer number of tags make the short data frames a wise choice for this application. But in other settings, such as checking the authenticity of high-value goods like pharmaceuticals, we can have the luxury of communicating with fewer tags at a time, for longer durations. This fact means we can appeal to the commands in Class-1 Gen-2 with variable-length data payload. To implement a security protocol, we will have to reuse commands designed for another purpose, or define custom or new commands.

This task of defining the use of one protocol to carry the protocol data units of another is often called *protocol convergence.* To refer to data units consumed by a protocol entity not contained in the Class-1 Gen-2 spec, we will use the phrase *application protocol data units*, or APDUs.

Section 6.3.2.1 of the standard specifies four banks of memory which may be read or written by a reader: reserved, EPC, TID, and User. The User bank offers the most flexibility, allowing user-defined organization of arbitrary amounts of memory arranged in 16-bit words. Subject to some conditions possibly involving the presentation of a fixed password, the tag is obliged to obey Read or BlockWrite commands. But the contents of memory need not be fixed: neither the standard nor the conformance document [13] prohibit the manipulation of memory by logic in the tag. In fact, we could view the situation as interprocess communication implemented by shared memory. The reader writes data to a particular memory location in the tag. Logic in the tag reads from this location, processes the data frame, and writes its response to that (or a different but commonly agreed-upon) memory location. The reader obtains its result by reading from this memory location.

As a concrete example, consider a tag that has been singulated by a reader by successfully responding to a sequence of Query, ACK, and Req_RN commands to arrive in the Access state [12]. Now the reader and tag can participate in a security protocol. We use a special block of shared memory in the User memory bank, starting with word zero to transfer the security protocol's APDUs between the reader and logic in the tag. Since the Class-1 Gen-2 protocol follows a reader-talks-first paradigm, the exchange begins with a BlockWrite command which writes the contents of the APDU to the shared memory, as shown in Table 9, found in the appendix along with all other frame formats referenced in this paper.

The tag's Class-1 Gen-2 interface writes the data to the appropriate location, and then transmits its normal reply to indicate success. We observe at this point that because protocol APDUs are meant for immediate consumption, rather than long-term storage, the contents of the shared memory can be stored in RAM instead of EEPROM. This allows the tag to use the time and power ordinarily used for writing nonvolatile storage for interpretation of, and response to, the APDU. As usual following command transmission, the reader broadcasts a continuous wave (CW) for up to 20 msec to power the tag and allow it to complete its operation. Additional logic in the tag uses this power and time to interpret the APDU, compute a response if necessary, and write its response to the same - or another previously agreed-upon - memory location. Once this is done, the tag sends its usual reply frame, which in this case indicates the tag has interpreted the APDU and a response is available. If processing a command takes longer than 20 msec, the response APDU prepared by the tag can indicate that processing has not yet completed.

The reader can now obtain its response by issuing a Read command. As before, we will assume that the special block of shared memory is located in the User memory bank and starts at word zero. This command frame is illustrated in Table 10.

Using this message sequence in principle allows us to implement virtually any protocol. Rather than overloading the Read and BlockWrite commands, we could define new commands with the same intent: a WriteGenericAPDU and ReadGenericAPDU could be assigned their own command identifiers without changing the basic approach. Of course, since the underlying logical layer follows a reader-talks-first paradigm, some protocols will work better than others. In order to handle APDUs originated by the tag, one could have the reader periodically use a read command to check if the contents of shared memory have changed. But this polling-based approach is unwieldy, so we instead look for existing protocols that fit nicely with the tools at hand.

5 A Natural Command Set: ISO 7816-4

This problem of authenticating a severely constrained device is not unique to supply chain applications: smart cards have long been used for authentication. Given the protocol convergence ideas articulated above, our enhanced tag looks more like a contactless smartcard and less like a traditional "license plate" RFID tag. So we aim to draw on the collective design and widespread implementation experience available in the smartcard arena to address our need for authentication and security feature negotiation. An ideal protocol would allow for extreme optimization of the most commonly used security features while also allowing other security operations possibly involving long APDUs fragmented into several data frames, and feature negotiation among cards and readers from different vendors.

We find such a protocol in part of ISO 7816 [16], a series of international standards that forms the basis for millions of smart cards worldwide including pay-TV and GSM SIM cards. As with many standards for communicating systems, the several documents in the ISO 7816 series are each devoted to a particular layer in a stack of protocols. This layered approach allows particular standards in the series to be applied to different environments. For instance, the ISO 14443 [15] series of standards for contactless proximity cards explicitly allows for the use of ISO 7816-4 APDUs to be carried over its logical and physical layers. From the perspective of a lower layer protocol, an ISO 7816-4 APDU would simply be seen as a data payload.

ISO 7816-4 offers a set of APDUs arranged in command-response pairs to authenticate and securely access data stored on a card. The specification declines to specify algorithms, physical interface technology, or the internal implementation within the card. Fortunately, most of its features are designed for systems where the reader talks first, nicely complementing the logical layer features in Class-1 Gen-2.

ISO 7816-4 defines general command and response frames, depicted in Table 1 and Table 2, respectively. It further specifies instantiations of these to perform tasks like entity authentication of tag, reader, or both as well as transfer of encrypted or integrity-protected data. To make things concrete, we'll focus on one command called Internal Authenticate, while our techniques extend to other commands as well.

Field	**Description**	**Number of bytes**
Command header	Class byte denoted CLA	1
Command header	Instruction byte denoted INS	1
Command header	Parameter bytes denoted P1-P2	2
Command data-length L_c	Absent if $N_c = 0$, otherwise equal to N_c	0, 1, or 3
Command data	Absent if $N_c = 0$, otherwise a string of N_c bytes	N_c
Maximum response length	Absent if $N_e = 0$, otherwise equal to N_e	0, 1, or 3

Table 1. ISO 7816-4 Command

Field	**Description**	**Number of bytes**
Response data	Absent if $N_r = 0$, otherwise a string of N_r bytes	N_r
Response trailer	Status bytes SW1 and SW2	2

Table 2. ISO 7816-4 Response

With this set of headers, data lengths, and trailers, the reader can unambiguously specify precisely which command is desired along with details on algorithms, protocols, parameters, key identifiers, and of course, command data. The tag can reply with status bytes indicating success, reasons for failure, or the fact that processing has not yet completed.

5.1 Features of ISO 7816-4

As noted above, we envision the Class-1 Gen-2 standard coming to serve as a radio interface for many applications beyond simple tagging. Fortunately, the ISO 7816-4 standard is highly developed with regard to interfacing with almost any security-capable device. But this capability goes beyond simple authentication to include important topics like supporting multiple algorithms, multiple keys, and multiple file structures. Of course, we cannot exhaustively describe all of these features in a paper such as this. Instead, in the appendix we enumerate some of the most important capabilities.

Given this rich feature set, one can address a great number of applications. But we observe that in this environment, tags may specialize on one or two security services such as authentication of a tag to a reader to prevent counterfeiting. In the rest of this paper, we focus on heavily optimizing a tag's most-used feature while still allowing the richness of the 7816-4 command set.

From the tables one can see there is some overhead associated with this command set: a typical command would see six bytes overhead, while a response would see two. Since communication bandwidth is precious in this environment, we must explore some examples to determine if the cost is acceptable and consider ways to reduce this cost.

5.2 Tag Authentication

In Section 4.1, we outlined a simple tag authentication protocol using challenge-response and one-time passwords. Using the techniques outlined in Section 4.2, we can go beyond this approach to support virtually any entity authentication protocol from simple passwords to robust cryptography. Let us consider the use of the ISO 7816 command set to achieve entity authentication of the tag.

Of course, we must choose some algorithm to achieve this goal. When it comes to cryptographic functions, we face an embarrassment of riches. Such a broad set of protocols, algorithms, and associated modes of operation has been devised that it seems shortsighted to attempt to fix one choice for all secure applications. Like the various options offered in the Class-1 Gen-2 physical layer, each of these cryptographic primitives conducts a careful trade off among attributes. In this case, the attributes are computational complexity, communication complexity, security services offered, and resistance to various types of attackers. We are forced then to choose one algorithm or devise some sort of negotiation scheme for a tag and reader to agree on a protocol, algorithms, and modes.

This service is precisely what ISO 7816 provides: security protocol messages tagged to reference an algorithm and any associated reference data such as a key identifier. By way of example, in Table 3, let us consider the Internal Authenticate command to implement our protocol. The value C_R will be provided by the reader in the protocol. Values in the table postfixed by "h" indicate hexadecimal notation.

Table 3 shows the reader providing an eight-byte challenge value to the tag. Note that the class and command parameter bytes are set to zero. Tables 2 and 3 in [16] define the semantics of the class byte. A reader can indicate if this command frame is a fragment of a longer command and if any encryption or integrity protection has been applied. In our case, neither of these conditions is true and therefore these bytes are set to zero. The command parameter identifies the algorithm,

Field	Description	Number of bytes
Command Class Byte	0h	1
Command Instruction Byte	88h	1
Command Parameters	0h	2
Command data-length	8h	1
Command data	C_R	8
Maximum response length	8h	1

Table 3. Internal Authenticate Command from Reader

Field	Description	Number of bytes
Response data	R_T	8
Status bytes	6100h	2

Table 4. ISO 7816-4 Response

protocol, and modes, but ISO 7816 allows these bytes to be set to zero if their values are implicitly known. For reasons of cost and efficiency, many tags may support only one set of these values.

When these APDUs are carried by Read or BlockWrite commands, we can calculate the total number of bytes sent over the air before our compression techniques in Section 5.3. By way of comparison, we also consider the case when the challenge, C_R, is implicitly known by the tag such as in time-synchronous onetime passwords.

Frame Type	Bytes	Bits
BlockWrite Carrying Internal Authenticate with Challenge	22	169
BlockWrite Carrying Internal Authenticate with Implicit Challenge	13	97
Read Carrying Response	18	137

Table 5. Frame sizes for shoehorned Internal Authenticate

5.3 Compressing ISO 7816-4

Clearly, these commands are larger than we would like. Our goal is to optimize the most common usage while allowing flexibility. Our working assumption is that most tags will support a small number of security methods and generally their usage will be implicit. This means in general that the class and parameter bytes - and sometimes the instruction byte - will be redundant. So we can eliminate these, but we need some way to signal to the tag which fields are present in a received data frame.

As above, we have two options: we can carry on using the Read and BlockWrite commands and specify a wrapper with a bit field to indicate which ISO 7816 fields are present. In essence, this wrapper becomes our security sublayer and allows the tag unambiguously reconstruct the original ISO 7816 APDU, if desired. As an alternative to Read and BlockWrite,we can define custom commands for this purpose. Both new and custom commands are considered below.

Security Sublayer Continuing our use of the Read and BlockWrite commands, we can prepend all ISO 7816-4 APDUs with a header to indicate which fields are present as shown in Table 6.

	Command Class Byte	Command Instruction Byte	Command Parameters
Number of bits	1	1	1

Table 6. Security Sublayer Header

Then a complete BlockWrite data frame to send an ISO 7816-4 APDU for an entity authentication protocol as above to compute the 64-bit value R_T given the 64-bit value C_R provided by the reader appears in Table 11, showing a savings of 29 bits compared with Table 5. As noted above, these parameters are provided as an example and many other combinations are possible, including an implied C_R value and a 32-bit password returned as in Table 12. Observe that ISO 7816-4 already allows the DataLen and Data fields to be omitted entirely if their values are implied, relieving us of the need to explicitly signal their presence. This result leaves us with a data frame of only 68 bits, 32 of which are the handle and CRC. Response frames are unchanged and remain as above. A summary of over-the-air complexity is in Table 7. But further reductions are possible if we turn to specialized commands.

Frame Type	Bytes	Bits
Compressed BlockWrite Carrying Internal Authenticate with Challenge	18	140
Compressed BlockWrite Carrying Internal Authenticate with Implicit Challenge	9	68
Read Carrying Response	18	137

Table 7. Frame sizes for Compressed Internal Authenticate using BlockWrite

New Commands To save even more bits over the air, we can turn to new commands. The standard defines command identifiers using up to 8 bits each for base commands, and 16 bits each for custom or proprietary commands. We observe that in our use of the Read and BlockWrite commands, quite a few bits are devoted to specifying a memory location and data length. A new or custom command's identifier would directly imply the memory location, saving some bits. In addition, the ISO 7816 APDU either specifies its own length explicitly in the DataLen field, or – like other parameters – it is previously known by both parties, allowing us to optionally dispense with the WordCount field. By defining a new command we can save a total of 17 bits by using an unreserved 8-bit identifier, of which there are 22 currently available. Of course, we could define a custom command instead, but then we would only save 9 bits since an additional 8 bits are required to specify a custom command. The command and response versions are illustrated in Tables 13-16.

Using this approach, we can compare the size of the data frames in each of these scenarios when used with our example cryptographic protocol in Table 8.

Authentication of a Group of Tags So far, we have focused on commands that require a tag to be fully singulated before a cryptographic protocol can take place. Recalling our example security

Frame Type	Bytes	Bits
New EPC-layer Command for ISO 7816 APDU Command with Challenge	15	123
New EPC-layer Command for ISO 7816 APDU Command with Implicit Challenge	6	51
Custom EPC-layer Command for ISO 7816 APDU Command with Challenge	16	131
Custom EPC-layer Command for ISO 7816 APDU Command with Implicit Challenge	8	59
EPC-layer Tag Reply to Response APDU	15	113

Table 8. Sizes of New or Custom Commands

function $R_T = H(K_{TS}, C_R)$, we observe that if each tag has a unique key K_{TS}, then the value C_R need not be unique to each tag. In fact, this is common practice in the application of one-time passwords for user login as C_R will often be either the time of day or a counter. Performing entity authentication of a group of tags can be greatly optimized by delivering C_R to all tags and allowing them to respond individually. Toward this end, we propose QuerySecure and ACKSecure commands which perform these functions using the protocol convergence ideas outlined above. Essentially, QuerySecure extends Query by appending the Header, DataLen, Data, RespLen, and 7816 APDU fields and replacing the CRC-5 with a CRC-16. The resulting data frame weighs in at 101 bits, but in contrast to the commands listed above, it only needs to be sent once to a population of tags. The ACKSecure command likewise simply appends the Response Data and Status Bytes fields to the existing ACK command which then scrolls back its EPC followed by the R_T value it computed.

6 Conclusion

The optimizations we have presented here are driven by the desire to optimize one commonly used cryptographic operation for each tag, while allowing the flexibility of a fully extensible, broadly supported, and internationally recognized protocol to handle issues of feature negotiation.

This is the motivation behind our fusing the EPC standard with ISO 7816-4. In the case of tag authentication using a block cipher, the resulting optimized data frames are shorter than many Electronic Product Codes. An implementor is not restricted to our example cryptographic protocol, or even to the ISO 7816-4 Internal Authenticate command. By either inspecting or knowing the tag's TID, the reader can use whichever 7816-4 command and associated parameters for which the tag is optimized. To access other commands, the reader can explicitly specify the desired command and parameters. We have shown three different ways to add this functionality to the Class-1 Gen-2 standard while maintaining backward compatibility: by using the BlockWrite and Read commands, by defining custom commands, and by defining new commands. An implementor could choose whichever of these methods is most suited to a particular deployment.

In summary, our proposed techniques permit the creation of RFID tags that are compliant with the Class-1 Gen-2 EPC standard, but offer the broad and widely supported cryptographic functionality of standards like ISO 7816-4. We hope that the simplicity and ready extensibility of our techniques will pave the way for the penetration of EPC into a broader array of security applications. EPCglobal has expressed the intention to create a Class-2 standard that specifies higher-functionality, higher-security, next generation EPC tags. Our approach could make this a much easier job, and allow also a broad spectrum of new devices to benefit from the infrastructure of today's EPC standard.

References

1. William A. Arbaugh, Narendar Shankar, and Y.C. Justin Wan. Your 802.11 wireless network has no clothes. Referenced 2005 at http://citeseer.ist.psu.edu/arbaugh01your.html.
2. G. Avoine, E. Dysli, and P. Oechslin. Reducing time complexity in RFID systems. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography – SAC 2005*, Lecture Notes in Computer Science. Springer-Verlag, 2005. To appear.
3. S. Bono, M. Green, A. Stubblefield, A. Juels, A. Rubin, and M. Szydlo. Security analysis of a cryptographically-enabled RFID device. In P. McDaniel, editor, *14th USENIX Security Symposium*, pages 1–16. USENIX, 2005. Dedicated Web site at www.rfidanalysis.org.
4. J. Collins. Ge uses RFID to secure cargo. *RFID Journal*, 12 January 2005. Referenced 2006 at http://www.rfidjournal.com/article/articleview/1317/1/1/.
5. T. Dierks and C. Allen. The tls protocol version 1.0. Referenced 2005 at http://www.ietf.org/rfc/rfc2246.txt.
6. ECRYPT (European network for excellence in cryptology), stream cipher project Web page, 2005. Referenced 2005 at http://www.ecrypt.eu.org/stream/.
7. M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In M. Joye and J.-J. Quisquater, editors, *Workshop on Cryptographic Hardware and Embedded Systems CHES 04*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370. Springer-Verlag, 2004.
8. K. P. Fishkin, S. Roy, and B. Jiang. Some methods for privacy in RFID communication. In *1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004)*, 2004.
9. United States Food and Drug Administration. Combatting counterfeit drugs: A report of the Food and Drug Administration, 18 February 2004. Referenced 2005 at http://www.fda.gov/oc/initiatives/counterfeit/report02_04.html.
10. P. Golle, M. Jakobsson, A. Juels, and P. Syverson. Universal re-encryption for mixnets. In T. Okamoto, editor, *RSA Conference - Cryptographers' Track (CT-RSA)*, volume 2964 of *Lecture Notes in Computer Science*, pages 163–178, 2004.
11. IEEE. Ieee 802.11i-2004 amendment to ieee std 802.11, 1999 edition (reaff 2003). ieee standard for information technology–telecommunications and information exchange between system–local and metropolitan area networks specific requirements–part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications–amendment 6: Medium access control (mac) security enhancements. Referenced 2005 at http://standards.ieee.org/getieee802/download/802.11i-2004.pdf.
12. EPCglobal Inc. Class 1 generation 2 UHF air interface protocol standard version 1.0.9. Referenced 2005 at http://www.epcglobalinc.com/standards_technology/EPCglobalClass-1Generation-2UHFRFIDProtocolV109.pdf.
13. EPCglobal Inc. Class 1 generation 2 UHF rfid conformance requirements version 1.0.2. Referenced 2005 at http://www.epcglobalinc.com/standards_technology/EPCglobalClass-1Generation-2UHFRFIDConformanceV102.pdf.
14. Texas Instruments and VeriSign, Inc. Securing the pharmaceutical supply chain with RFID and public-key infrastructure technologies. Whitepaper. Referenced 2005 at http://www.ti.com/rfid/docs/customer/eped-form.shtml.
15. ISO. Identification cards – contactless integrated circuit(s) cards – proximity cards – part 4: Transmission protocol. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=31425.
16. ISO. Identification cards – integrated circuit cards – part 4: Organization, security and commands for interchange. Referenced 2005 at http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=36134.
17. A. Juels. Minimalist cryptography for low-cost RFID tags. In C. Blundo and S. Cimato, editors, *The Fourth International Conference on Security in Communication Networks – SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 149–164. Springer-Verlag, 2004.
18. A. Juels. 'Yoking-proofs' for RFID tags. In R. Sandhu and R. Thomas, editors, *Workshop on Pervasive Computing and Communications Security – PerSec 2004*, pages 138–143. IEEE Computer Society, 2004.
19. A. Juels. Strengthing EPC tags against cloning. In *ACM Workshop on Wireless Security (WiSe)*. ACM Press, 2005. To appear.
20. A. Juels. RFID security and privacy: A research survey. *J-SAC*, 2006. To appear. Online version referenced 2005 at http://www.rsasecurity.com/rsalabs/node.asp?id=2937.
21. A. Juels, D. Molnar, and D. Wagner. Security and privacy issues in e-passports. In D. Gollman, G. Li, and G. Tsudik, editors, *IEEE/CreateNet SecureComm*. IEEE, 2005. To appear. Referenced 2005 at http://www.cs.berkeley.edu/ dmolnar/papers/papers.html.

22. A. Juels, R.L. Rivest, and M. Szydlo. The blocker tag: Selective blocking of RFID tags for consumer privacy. In V. Atluri, editor, *8th ACM Conference on Computer and Communications Security*, pages 103–111. ACM Press, 2003.
23. A. Juels, P. Syverson, and D. Bailey. High-power proxies for enhancing RFID privacy and utility. In G. Danezis and D. Martin, editors, *Privacy Enhancing Technologies (PET)*, 2005. To appear.
24. A. Juels and S. Weis. Authenticating pervasive devices with human protocols. In *Advances in Cryptology – CRYPTO 2005*, pages 293–308. Springer-Verlag, 2005. Lecture Notes in Computer Science, Volume 3621.
25. G. Karjoth and P. Moskowitz. Disabling RFID tags with visible confirmation: Clipped tags are silenced (short paper). In S. De Capitani di Vimercati and R. Dingledine, editors, *Workshop on Privacy in the Electronic Society (WPES)*, 2005. To appear.
26. D. Molnar, A. Soppera, and D. Wagner. Privacy for RFID through trusted computing (short paper). In S. De Capitani di Vimercati and R. Dingledine, editors, *Workshop on Privacy in the Electronic Society (WPES)*, 2005. To appear.
27. D. Molnar and D. Wagner. Privacy and security in library RFID : Issues, practices, and architectures. In B. Pfitzmann and P. McDaniel, editors, *ACM Conference on Communications and Computer Security*, pages 210 – 219. ACM Press, 2004.
28. M. Ohkubo, K. Suzuki, and S. Kinoshita. Efficient hash-chain based RFID privacy protection scheme. In *International Conference on Ubiquitous Computing – Ubicomp, Workshop Privacy: Current Status and Future Directions*, 2004.
29. R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical one-way functions. *Science*, 297:2026–2030, 20 Sept. 2002.
30. M. Rieback, B. Crispo, and A. Tanenbaum. RFID Guardian: A battery-powered mobile device for RFID privacy management. In Colin Boyd and Juan Manuel González Nieto, editors, *Australasian Conference on Information Security and Privacy – ACISP 2005*, volume 3574 of *Lecture Notes in Computer Science*, pages 184–194. Springer-Verlag, 2005.
31. T. Staake, F. Thiesse, and E. Fleisch. Extending the EPC network – the potential of RFID in anti-counterfeiting. In *ACM Symposium on Applied Computing*, pages 1607–1612. ACM Press, 2005.
32. P. Tuyls and L. Batina. Rfid-tags for anti-counterfeiting. In *CT-RSA 06*. Springer-Verlag, 2006. To appear.
33. I. Vajda and L. Buttyán. Lightweight authentication protocols for low-cost RFID tags. In *Second Workshop on Security in Ubiquitous Computing – Ubicomp 2003*, 2003.

A Security Services

A.1 Device authentication

EPC tags and other low-cost RFID devices are often referred to as “license plates:” They carry and broadcast fixed identifiers. In consequence, such devices are easy to *clone*. An attacker can read an identifier and write it to a new, programmable RFID tag or simulate it in a different type of RF device. Mitigating the risk of tag cloning is an essential security goal in an RFID system. Cryptographic authentication can help achieve this goal. Tags that possess secret keys and execute well-designed cryptographic protocols to authenticate themselves to readers can resist cloning in the face of over-the-air attack.

Conversely, it may also be desirable for tags to be able to authenticate readers. Such authentication can help restrict the availability of sensitive tag data to unauthorized readers.

Of course, cryptographic protocols are effective only in logical-layer defense. Keys are also subject to physical compromise. An attacker that physically probes an RFID tag and extracts its secret keys can readily clone it. While the cryptographic services that we propose here cannot directly forestall such attacks, they can support tamper-resistance mechanisms, like PUFs and POWFs [32, 29], that rely on a blend of logical and physical countermeasures.

A.2 Device-binding authentication

RFID tags serve to identify the objects or people that bear them. Thus, even if a tag is valid, i.e., has not been cloned, it may still furnish erroneous information if inappropriately placed. It is important, therefore, to establish the correctness of the physical context for an RFID tag. Toward this end, a range of physical mechanisms has been proposed. For example, shipping containers have been designed that contain internal RFID devices whose state changes in response to the opening of the container [4]. PUFs [32] and POWFs [29] are physical objects – silicon and glass-and-plastic respectively – whose state changes in response to physical stresses, and can help in the detection of RFID-tag removal. (Of course, good adhesives can also help.) Various chemical fingerprinting and watermarking techniques already help combat counterfeiting, and can work in harmony with RFID devices.

A.3 Data-origin authentication

An RFID tag can, of course, serve not just as an identifier, but also as a carrier of ancillary data, e.g., information about goods in a pallet to which it is attached. For this reason, data-origin authentication also plays a role in RFID security. Indeed, data-origin authentication can support the physical integrity of the tag itself, as when an RFID tag stores information about the state of physical tamper-detection objects like PUFs or POWFs or chemical markers.

As a cryptographic service, data-origin authentication binds the production of a block of data to a particular entity by means a digital signature (or message-authentication code (MAC)). In addition, this service attests that the data have not been modified since their original encoding. An RFID tag can carry data digitally signed by an external entity. In this case, the tag serves merely as a data carrier, and need not itself perform the cryptographic operation of signing. A high-powered RFID tag can alternatively compute a digital signature on data it produces dynamically.

A.4 Example: Pharmaceutical tagging

To illustrate the roles that various types of authentication play in RFID security, let us consider the use of RFID in the pharmaceutical supply chain. The United States Food and Drug Administration has advocated such use of RFID as an anti-counterfeiting tool [9]. Its deployment over the next couple of years seems all but certain, and EPC tags, given their cheapness and ubiquity, are likely to be the RFID device of choice.

Texas Instruments (TI) and VeriSign [14] have proposed an architecture for pharmaceutical applications in which tags serve as carriers of digitally signed "event" data, timestamped records whose contents are unspecified (but "do not contain any product or location information"). In other words, the system provides data-origin authentication. In principle, the system provides assurance that the information a reader obtains from a tag has been created by authorized readers within the system and not fabricated by an attacker.

The system (as specified in [14]) does *not* provide full device authentication. In particular, it offers only weak assurance against the cloning of a tag. (Tags contain "locked" but static identifiers.) Consider, for example, a system in which pallets of medications of type A or type B are shipped from either site X or site Y to site Z. A pallet might bear a tag containing a statement S digitally signed by site Y that, "This is pallet #12345, carrying a medication of type A shipped from site Y to site Z." If the tag has been displaced or cloned, however, a reader cannot be assured that the tag in question is attached to the correct pallet. The pallet might in fact contain medication of type B shipped from site X, and the tag might have been cloned from a different pallet!

Device authentication, and authentication of RFID tags, can help mitigate this problem. If tags in the system do not merely bear digitally signed data, but perform cryptographic authentication, then a reader can achieve assurance that the tag with which it is communicating not been cloned. For example, the RFID tag on pallet #12345 might use a challenge-response protocol to prove the correctness of its digital identifier, say, "ID67890123".

Such authentication is still not sufficient, however, to ensure the correctness of data received by the reader. How, for instance, does the reader know that the statement S carried by tag ID67890123 was written to the tag by an authorized reader, and not simply copied from some other tag? (The TI/VeriSign architecture specifies that S carry the tag ID, in order to establish such a binding.) A problem still remains, however. How can a reader achieve assurance that a tag has not simply been peeled off one pallet and attached to another one?

To address this problem, device-binding authentication is required. Suppose that pallet #12345 bears the number "12345" in a form that is difficult to duplicate or alter, such as a hologram. Then the digital signature S, which contains both the pallet number and transaction data, helps ensure a correct association between the transaction data and the pallet with which the data are associated. (The TI/Verisign paper refers to a forthcoming paper that will describe device-binding authentication.)

At first glance, device-binding authentication and data-origin authentication alone might seem to obviate the need for device authentication. This is not necessarily the case. For example, device-binding authentication as we have described it might require optical scanning of a pallet. If a pallet must be optically scanned, then the benefits of RFID, namely readability without line-of-sight contact, are eliminated. An RFID tag that performs cryptographic authentication offers a heightened degree of system integrity – if not full integrity – without burdensome scanning procedures. Similarly, it may be that the information contained in S is sensitive, and should only be accessible to authorized readers. To achieve strong access control on S, a tag would need to perform cryptographic authentication of interrogating readers. Finally, the problem of key management must be taken into account. If tags do not authenticate themselves, then the system relies for data integrity on reader keys alone. This situation necessitates the presence of multiple signing keys (with the associated management and storage overhead), or a single, monolithic signing key representing a single point of compromise.

The interplay among cryptographic (and other) authentication services is complex. We do not purport to offer a comprehensive solution here to the problem of counterfeiting. Our example here simply illustrates the need for a flexible range of cryptographic services for pharmaceutical supply chains, as for the many other emerging applications of EPC tags.

B Features of ISO 7816-4

B.1 File-Oriented Commands

ISO 7816 offers a rich set of commands to handle various filesystems beyond the bitmapped flat-file system offered in the Class-1 Gen-2 standard. In particular, a tag can support multiple files arranged in linear or cyclic structure as a mechanism to support multiple applications in the supply chain. For instance, the various entities handling an article in a supply chain could have their own data file to use as they see fit. This approach can enable granular access control. In principle, each file could be encrypted using a key known only to one or a small subset of supply chain participants. Or an item's entire pedigree could be stored on the tag in a series of files each digitally signed by the supply chain participant responsible for that file.

B.2 Multiple Applications

This ability to support multiple applications extends not only to the filesystem but also to the communicating entity. Imagine a tag affixed to military materiel offering multilevel security. In this scenario, one may want a restricted tag application which makes some logistics information and authentication capability available while a tag is in transit. By necessity, this feature may have to use unclassified algorithms and symmetric keys which are widely shared among supply chain participants. The tag's full functionality including its pedigree could be available to a full-featured tag application only available to authenticated readers using classified algorithms and keys closely held in hardware security modules.

B.3 Multiple Keys and Ciphersuites

Should a reader authenticate a tag? Should a tag authenticate a reader? Should they both authenticate each other? Which algorithm(s) should they use? It all depends on the application with some uses more concerned about anti-counterfeiting and others confidentiality. In ISO 7816, each application includes "one or more cryptographic mechanism identifier templates" to unambiguously specify the algorithm a reader should use for that application. This level of algorithm agility allows different entities to choose their own keys and security methods as suggested above in the multilevel security tag.

B.4 Authenticated Access Control

Beyond authentication, ISO 7816 includes features to restrict access to particular commands such as "Read Binary," "Delete File," and "Create File" For instance, reader authentication can be required before a command. In addition, an access control rule can specify that these commands and their responses must be encrypted and/or authenticated.

B.5 Security-Descriptive Data Exchange

How does a reader know it is receiving encrypted information? How does the reader know which algorithm, mode, and key were used? ISO 7816 includes tags to describe cryptograms.

B.6 Command Fragmentation

If the length of an ISO 7816-4 APDU stretches beyond the limits of the underlying data transport, it includes a facility to allow the APDU to be sent in pieces and reconstructed unambiguously.

B.7 Broad Existing Infrastructure

Perhaps most important, ISO 7816 is already implemented in millions of smart cards and readers including GSM phones. This wide availability of devices implies existing software tools which can be applied to this area.

C Frame Formats

	Command	**MemBank**	**WordPtr**	**WordCount**	**Data**	**Handle**	**CRC-16**
Number of bits	8	2	EBV	8	Variable	16	16
Description	11000111	11	0000000	Number of words to write	APDU	handle	

Table 9. Using BlockWrite command to carry an APDU

	Command	**MemBank**	**WordPtr**	**WordCount**	**Handle**	**CRC-16**
Number of bits	8	2	EBV	8	16	16
Description	11000010	11	0000000	Number of words to read	handle	

Table 10. Using Read command to carry an APDU

EPC Layer	**Cmd**	**Bank**	**Ptr**	**Count**	**Data**				**Handle**	**CRC**
Security Layer					Header	DataLen	Data	RespLen		
Number of bits	8	2	EBV	8	3	8	64	8	16	16
Description	11000111	11	0000000	00000110	000	00001000	C_R	00001000	handle	

Table 11. Using BlockWrite command with explicit challenge value

EPC Layer	**Cmd**	**Bank**	**Ptr**	**Count**	**Data**				**Handle**	**CRC**
Security Layer					Header	DataLen	Data	RespLen		
Number of bits	8	2	EBV	8	3	0	0	8	16	16
Description	11000111	11	0000000	00000110	000			00000100	handle	

Table 12. Using BlockWrite command with implicit challenge value

	Command	**Header**	**Compressed 7816 APDU**	**Handle**	**CRC-16**
Number of bits	8	3	Variable	16	16
Description	11001001		C_R	handle	

Table 13. New EPC-layer Command for ISO 7816 Command APDU

	Header	**Handle**	**CRC-16**
Number of bits	1	16	16
Description	0	handle	

Table 14. EPC-layer Tag Reply to ISO 7816 Command APDU

	Command	**Handle**	**CRC-16**
Number of bits	8	16	16
Description	11001010	handle	

Table 15. New EPC-layer Command for ISO 7816 Response APDU

	Header	**Response Data**	**Status Bytes**	**Handle**	**CRC-16**
Number of bits	1	Variable	16	16	16
Description	0	R_T			

Table 16. EPC-layer Tag Reply for ISO 7816 Response APDU