Tag Group Authentication Using Bit-Collisions

X. Leng
Department of Industrial Engineering
College of Engineering, Peking University, China
Email: xuefeileng@gmail.com

G.P. Hancke, K.E. Mayes and K. Markantonakis
ISG Smart Card Centre, Royal Holloway, University of London, UK
Email: gerhard.hancke/keith.mayes/k.markantonakis@rhul.ac.uk

***Abstract*—Secure RFID systems should be able to authenticate individual tags to prevent genuine items from being replaced with forgeries. Conventional authentication protocols would require some form of challenge-response exchange with each individual tag. The extra transaction time needed to authenticate each individual tag sequentially might not be practically feasible, especially when readers are required to process an ever increasing volume of tagged items within a limited time. Our proposal works towards an approach that allows for multiple tags in a group to be authenticated simultaneously, as all the tags transmit their responses to a single challenge at the same time. This results in a controlled bit-collision pattern that can be used to verify that all the individual tags in a group are present and genuine.**

I. Introduction

Radio Frequency IDentification (RFID) is expected to serve as a key component of future ubiquitous computing environments, with the technology already used in various e-ID, payment and supply chain services. As the uses of RFID have grown the security issues surrounding this technology have received increasing attention [1]–[3]. One of the primary security concerns in a suppply chain would be that attackers could create clones of existing tokens. This could allow them to replace valuable items in shipments with forgeries, which would appear like genuine products to the system's readers. In this case, authenticating the tags could aid in detecting and discouraging the use of cloned tags. Conventional authentication mechanisms, however, would require the reader to challenge each individual tag and wait for a response. Taking the time to run an authentication protocol with each tag in a high-volume supply chain system might not be a practical option, especially if there are a large number of individual tags that need to be read within a short period of time.

The time taken to authenticate a group of tags would be significantly reduced if multiple tags could be authenticated simultaneously instead of sequentially. We propose an authentication protocol contrcution that allows for multiple tags in a group to be authenticated using a single response, thus achieving an execution time comparable to that of a transaction with a single tag. The proposal builds on principles introduced by cryptograhpic 'grouping-proofs' and efficient tag discovery algorithms. It verifies that the a group is both complete (no tag is missing) and pure (no tag has been replaced), using only a keyed pseudo-random function and simple bit permutation operations. We also provide initial comments on the security performance and practical implementation considerations of our protocol.

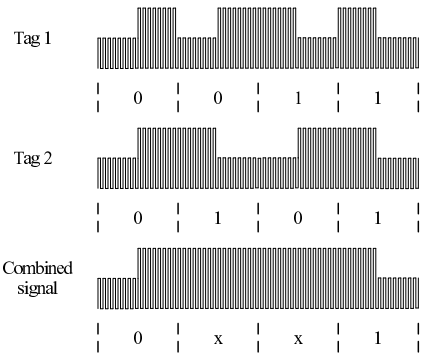

Fig. 1. Constructing bit-collision patterns with Manchester code

II. Background and Related Work

The idea of cryptographically verifying that a group of items is complete and pure has been discussed before. Juels first proposed the notion of a yoking-proof (yoke meaning "to join together") in which two tags can prove to the system that they are both present within the same reader's reading range at approximately the same time [4]. The proposal takes into account that passive RFID tags cannot directly communicate with each other, so to generate a proof the reader forwards messages from one tag to the other. Subsequently, several research papers have suggested improvements to Juels' proposal, e.g [5], [6]. Bolotnyy *et al.*presented a general implementation allowing a proof to be constructed for more than two tags [7], which was followed by further papers extending the yoking proof proposal to construct grouping or so-called 'co-existence' proofs for multiple RFID tags, e.g. [8], [9]. Burmester *et al.* [10] also presented a security model for grouping proofs and presented three grouping-proof protocols that their model showed to be provably secure. There are numerous further examples of grouping proofs in literature that are not included here due to space constraints.

Although grouping proofs enable the verifier to check that the group is complete and pure none of these proposals solve the transaction time problem. Some grouping proofs are often described as methods for proving that multiple tags were read "simultaneously", but in reality this only means that all the tags were queried sequentially within a specified time bound. For example, in a minimalist basic two-tag yoking proof both tags are within the reader's range at the same time, but the reader communicates separately with the two tags [4].

There are several proposals for reliably and efficiently reading a large number of RFID tags, e.g. [11], [12]. However, in these proposals a system only really determines if a tag is physically missing, i.e. a specific unique identifier (UID) within a group could not be read, and in some case multiple iterations of a read process is required. In security terms this is not good enough to differentiate between a real and a fake tag. The problem addressed in our paper is efficient *authentication*. Ideally all the tags in a group should be authenticated simultaneously using only a single challenge-response sequence.

In HF systems adhering to the ISO 14443 [13] and ISO 15693 [14] (ISO 18000-3 [15]) standards bit collisions are used during tag selection. The tokens' are synchronised and respond with Manchester-coded data, resulting in observed bit values and bit collisions, as shown in Figure 1. Deterministic bit-collisions have been incorporated into key exchange protocols [16], [17] and into security mechanisms providing confidentiality and privacy services by intentionally blocking tag responses to unauthorised readers [18], [19] but have to date not been used in authentication schemes. In all these proposals it is assumed that if a collision is observed then the attacker cannot determine who sent which bit symbol, e.g. which tag sent a '1' and which tags sent a '0'. In [20] it is shown that an attacker could in some cases deduce the responses from two tags contributing to a bit-collision pattern because of variations in the communication channel of passive tags. The chances of the attacker successfully using this method decreases, however, as the number of tags contributing to the collision pattern increases.

III. Using Bit-Collision Patterns for Group Authentication

RFID is widely used in automatic identification and data capture (AIDC) systems. The identified items can often be grouped into logical groups, e.g. a set of spare parts for a specific car or different medicines constituting a patient's prescription. In secure tracking applications the system often has to verify that all the items in the group are still together and that the items are genuine, i.e. that none of the items have been lost or replaced. This involves the system authenticating each tag, which is essentially the verifier issuing a challenge to each tag and the tag replying with a response that only it should know. For a large number of items the total time needed to run an authentication protocol with each tag might become impractical as the transaction time in RFID systems are generally limited. For example, RFID-tagged items might be rapidly moving down a conveyor belt or contained in a truck driving past the reader, so the tags will only be in the reader's field for a short period of time. As a result the time the reader has to communicate with each tag is restricted and a more efficient way of authenticating a group of tags is therefore needed.

Our scheme is based on the basic principle that if n tags in a group transmit their authentication responses at the same time this will result in a verifiable bit collision pattern that represents the authentication response for the entire group. To explain this further we need to define the bit collision operation more formally. We denote the collision operation between two bit symbols, β and β', $\beta, \beta' \in \{0, 1\}$ as $\beta \wedge \beta'$ and use x to indicate that a bit collision occurred. The results of $\beta \wedge \beta'$ are as follows:

β	β'	$\beta \wedge \beta'$
0	0	0
0	1	x
1	0	x
1	1	1

If two tags transmit 1 and 0 in the same bit slot it will result in a bit collision x regardless of the values transmitted by any other tags, i.e. $1 \wedge x = x$ and $0 \wedge x = x$.

In our proposed scheme each tag t_i, $i = 1$ to n, that is part of the group contains an authentication state s_i. If all the tags transmit their authentication states simultaneously then this will result in a group authentication state S, i.e. $S = s_1 \wedge s_2 \wedge \ldots \wedge s_n$. The group authentication state S can then be used to determine if the group is both complete and pure. If each tag in the group is configured to contribute at least one bit collision, and all tags cause an equal number of bit collisions, a verifier can check for completeness of the group. The verifier simply counts the number of bit collisions that occur. If there are less than it expects, it knows that either a tag is missing or a fraudulent tag has caused collisions at the same time as another tag that is present. If there are also a suitable number of bit positions where a collision does not occur than the verifier can check for purity. The verifier checks in which bit positions the collisions occur. If a bit collision is detected in a bit position where it was not expected the verifier knows that a fraudulent tag has failed to calculate in which bit position it should cause a collision or inadvertently caused an extra collision because it failed to guess a correct non-collision value. The verifier therefore knows that the group has been "contaminated". For example, if we have four tags each contributing one collision to a group authentication state of length 8 the authentication process works as follows:

tags	correct	missing s_4	fake s_4
s_1	01000011	01000011	01000011
s_2	01001001	01001001	01001001
s_3	01100001	01100001	01100001
s_4	11000001	missing	01010001
S	$x1x0x0x1$	$01x0x0x1$	$01xxx0x1$

If the group is complete and pure, i.e. all the tags are present and have the correct state as shown in the left column, then the verifier would expect four collisions in bit positions 1, 3, 5 and 7 respectively. If the groups were not complete, for example s_4 is missing, there would only be three bit collisions, as showed in the middle column, and the verifier will detect that a tag is missing. Similarly, if one tag (s_4 in this example) was replaced with a forged tag transmitting the incorrect state it would be detected, as the verifier would notice that no collision occurred in bit position 1 but rather in bit position 4.

A. Authentication State Initialisation and Permutation

The processes of initialising and updating the group authentication state are essential to the correct operation of our proposal. Tags cannot simply respond with independent, random responses as that would not represent the group in a way which would indicate whether the group is complete and pure. The authentication state must therefore be created and updated in a controlled manner that preserves the group's authentication information. We now examine in more detail how the initial authentication state S for each tag and the state permutation function f_2 could be constructed.

Group initialisation: For the purpose of proving completeness and purity at least one bit collision must be attributed to each of the n tags in the group. If this is not the case the tags that cause no bit collisions could be removed from the group without affecting the group authentication state. Each tag should contribute an equal number of bit collisions, which would allow the verifier to determine how many tags are missing if the group is found to be incomplete. Consider a group containing two tags, A and B, contributing one collision each and two tags contributing two each, C and D. As the tags are not contributing the same number of bit collisions the verifier does not know whether the group is missing two tags ($A+B$) or one tag (either C or D) if the group authentication state is found to be missing two bit collisions. A bit collision should also be paired with a bit value (non-collision) so that the bit swap operation in f_2 (see the following section) always changes the position of a bit collision, i.e. if a bit pair consisted of two collisions or non-collisions a bit swap would not cause a change in the collision pattern. If each tag contributes c bit collisions, which are each paired with a non-collision bit value, then the bit length of the group authentication state S and the individual tags' states s_i should be $2cn$. To meet all these criteria, we propose that the sender constructs the tag group authentication state as follows:

(1) Choose the number of collisions c that each of the n tags in the group will contribute.

(2) Create a state matrix M with n rows, each containing a tag authentication state vector of length $2cn$ and all values set to 0.

(3) Randomly select c bit pairs, $b_1, \ldots, b_c$, out of the set of cn possible pairs. In the first row set the first bit value of each chosen pair equal to 1, i.e. $M_{1,2b_i-1} = 1$ for $i = 1, \ldots, c$. Remove the previously selected $b_1, \ldots, b_c$ from the set, choose another c pairs from the remaining $c(n-1)$ pairs and set the corresponding bit values in row 2 to 1 in a similar way as before. Repeat until all n rows contain c collisions. For example, if $n = 4$ and $c = 2$ the tags' authentication states are set as follows:

	Tag states	Choosing bit pairs
s_1	1000000000001000	$1, 7$ of $(1, 2, 3, 4, 5, 6, 7, 8)$
s_2	0010100000000000	$2, 3$ of $(2, 3, 4, 5, 6, 8)$
s_3	0000001000100000	$4, 6$ of $(4, 5, 6, 8)$
s_4	0000000010000010	$5, 8$ of $(5, 8)$
S	$x0x0x0x0x0x0x0x0$	

(4) Load each tag with the group ID ID_G, group key k_g and set the sequence counter. It would be preferable if the initial value of the group's sequence number is randomly chosen so that the same sequence numbers are not repeated in authentication transactions with different groups.

(5) Finally, the sender runs the group authentication protocol with the tags to randomise the values of the non-collision bits and the positions of the bit collisions before shipping.

State permutation function f_2: f_2 should be chosen to satisfy the following property: $f_2(s_1) \wedge f_2(s_2) \wedge \ldots \wedge f_2(s_n) = f_2(S)$. The simplest way to do this is to create a permutation function using XOR, bit swap and shift operations. The bit swap operation switches the collision and non-collision value in a bit pair if the corresponding bit in the swap string is '1'. The shift operation bit rotates the entire authentication state (the bits wrap around). The bit swap and shift operations do not effect the number of bit collisions or the tag which is responsible for a specific bit collision, but these operations do change the positions of the bit collisions. Assuming that the initial state is created using the method described above we always shift the string in multiples of 2 (or in bit pairs) so that a pair always consists of one bit collision and one non-collision value. This ensures that a bit swap will always change a collision position.

The XOR operation does not effect the number or positions of the bit collisions even though it might change the underlying bit values contributing to the collision, i.e. if a, b, c are binary bits, then if $a \neq b$, $(a \oplus c) \wedge (b \oplus c) = x$. This property always holds because if $a \neq b$ then $(a \oplus c) \neq (b \oplus c)$, which results in $(a \oplus c) \wedge (b \oplus c) = x$. The XOR operation's primary purpose is to randomly alter the non-collision bit values.

In the protocol, f_2 will accept the results of a keyed pseudo-random function f_1. If there are n tags, each contributing c collisions, then the length of the authentication states (s_i or S) is $2cn$ and there are cn bit pairs, which can be shifted between 1 and $cn-1$ positions. The result of the pseudo random function can be parsed into three bit strings that define the state permutation, XOR string ($2cn$ bits), bit swap string (cn bits) and shift string ($\text{ceil}(\log_2(cn))$ bits). The pseudo-random function should therefore yield a result of length $3cn + \text{ceil}(\log_2(cn))$ bits. If a single output of f_1 is too short a longer pseudo-random number could possibly be created by running the function repeatedly using the previous result as input.

The following example illustrates how the state permutation function works. If we have four tags with authentication states equal to

s_1	01000011
s_2	01001001
s_3	01100001
s_4	11000001
S_{old}	$x1x0x0x1$

and the pseudo-random function yields a bit string

$$\underbrace{0101}_{\text{Swap}}\ \underbrace{01}_{\text{Rotate}}\ \underbrace{10100101}_{\text{XOR}}$$

then we need to swap bit pairs 2 and 4, rotate right by one bit pair and XOR the result with 10100101.

	$Swap$	$Shift$	XOR
s_1	01000011	11010000	01110101
s_2	01001010	10010010	00110111
s_3	01010010	10010100	00110001
s_4	11000010	10110000	00010101
S_{new}	$x10xx01x$	$1xx10xx0$	$0xx10xx1$

From the example it can be seen that $S_{\text{new}} = f_2(S_{old}) = f_2(s_1) \wedge f_2(s_2) \wedge f_2(s_3) \wedge f_2(s_4)$.

B. Group-Authentication Protocol

This section describes how the group authentication state operation defined above could be incorporated into a authentication protocol. Our protocol proposal is based on the following assumptions about the environment of the RFID system:

- **System operation:** The system consists of a number of nodes that track the progress of a group of items, i.e. a single package or shipment, from its sender to the intended recipient. The sender, recipient and nodes could be controlled by different organisations, but a key management infrastructure is in place that allows the sender to distribute key material to the verifying nodes and the recipient. The group verifier, i.e. a node or the recipient, does not necessarily have the ability to share information with other verifiers and as a result it should not require knowledge of previous protocol runs between the group and other verifiers.
- **Security objectives:** The purpose of our protocol is only to prove the completeness and purity of a chosen group to the recipient and intermediate verifying nodes, thus preventing a third-party attacker from replacing or stealing items during shipping. The protocol does not provide non-repudiation of purity and completeness for anyone who does not trust the verifying node or recipient. The sender, recipient and verifying nodes are seen as trusted entities, who will not reveal key material or create fraudulent tags/groups.
- **Cryptographic primitives:** For the purpose of running the protocol a group of tags, sender, recipient and the verifying nodes share a dedicated secret group key k_g, a keyed public pseudo-random function f_1 and a public bit permutation function f_2. In practice $f_1(m, k_g)$ would probably be based either on symmetric encryption, e.g. $\text{Enc}(m)_{k_g}$, or a hash function, e.g. $\text{h}(m, k_g)$.
- **Grouping process:** A group is made up of a number of related items labeled with RFID tags. A group would be a number of items physically packaged together as these must be interrogated together by a single reader. The sender is responsible for creating a legitimate group and initialising the tags. The sender will also send the group ID and a description of the items to the intended recipient, via a suitable communication channel, to enable the recipient to identify the contents. If required, tags can contain additional data, e.g. an item description or serial number, protected with a key shared between the sender and recipient, although this is beyond the scope of our protocol.

Our protocol proposal is shown in Figure 2. As stated previously, the tags and the verifier share a keyed public pseudo-random function f_1. The tags and the verifier also share a state permutation function f_2, which is described in more detail in Section III-A. Each tag t_i, with $i = 1$ to n where n is the number of tags, contains at least a stored current authentication state $s_{i_{\text{current}}}$, a common counter value seq, a common group identifier ID_G and a common group key k_g. The protocol comprises of the following steps:

(1) The verifier signals its intention to start the authentication protocol by transmitting a *Start* command. The *Start* command could also be used to narrow down the groups responding, by transmitting a value similar to the Application Family Identifier (AFI) used by EPC (ISO 18000-6C) and ISO 15693/18000-3 tags.

(2) All the tags simultaneously respond with a group ID ID_G, counter value seq and the number of tags in the group n. If a bit collision occurs only in seq the verifier knows that a tag has become unsynchronised, while a bit collision in both seq and ID_G indicates that tags from another group have also responded. The verifier can use ID_G to select, or derive, the correct group key.

(3) Next the verifier generates a random bit string r and calculates $m_1 = f_1(r, seq, k_g)$. It then transmits the group ID ID_g of the tags it wishes to authenticate, the bit string r and m_1. All the tags in group ID_g now calculate $m_1' = f_1(r, seq, k_g)$ and verify that $m_1' == m_1$, thereby essentially authenticating the verifier. The combination of the random bit string r and seq provides freshness for each protocol run, which prevents an attacker from replaying previous authentication responses.

(4) If the verifier is shown to be legitimate the tags all transmit their current authentication state $s_{i_{\text{current}}}$, so that the verifier can learn the current composite group authentication state S_{current}. This is required for synchronisation between the group and the verifier as we assume that there is not necessarily a communication channel between verifiers that can be used to share the group's last known state. An exception is raised if no tags transmit their authentication state and the protocol information should be reported to the sender. This is to discourage the 'dummy' tag attack described within Case 4 in Section IV.

(5) The tags then each update their authentication state using the state permutation function f_2 and $s_{i_{\text{new}}} = f_2(s_{i_{\text{current}}}, m_2)$, where $m_2 = f_1(r, seq + 1, k_g)$. Subsequently, each tag transmits $s_{i_{\text{new}}}$ resulting in a new group authentication state S_{new}. If no responses are received from the tags a security exception is raised.

(6) In the meantime the verifier has calculated $S'_{\text{new}} = f_2(S_{\text{current}}, m_2')$, where $m_2' = f_1(r, seq + 1, k_g)$, so it can verify that all the tags updated their authentication state correctly by comparing $S_{\text{new}} == S'_{\text{new}}$, thereby effectively authenticating the group.

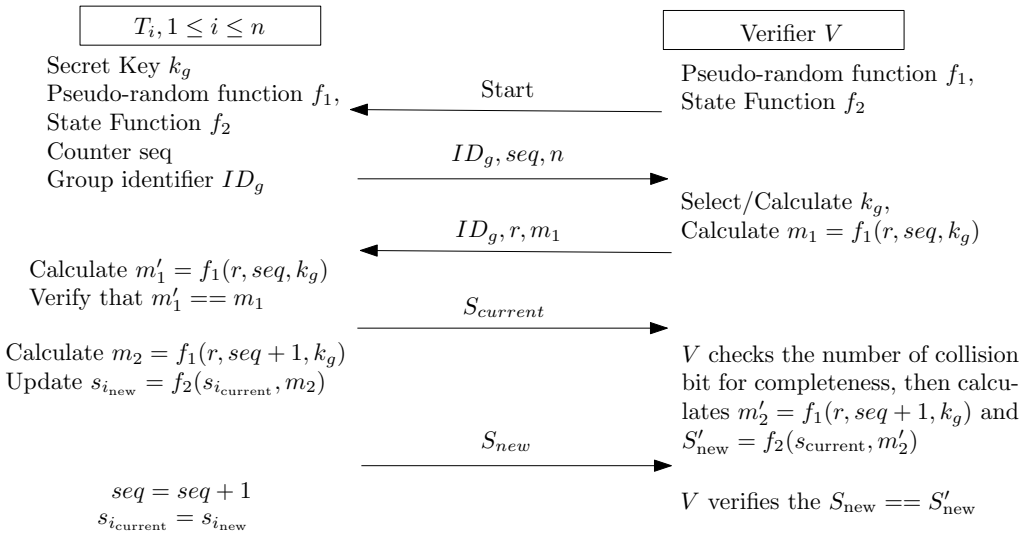

Fig. 2. Description of the proposed group-authentication protocol

(7) Finally, each tag increments seq and sets $s_{i_{\text{current}}} = s_{i_{\text{new}}}$. The verifier also reports the identifier ID_g, sequence number seq, the random bit string r and the new authentication state S_{new} to the sender. The sender learns the current location of the group, i.e. the group is in close proximity to a known verifying node, and stores the additional information for audit purposes.

If $S_{\text{new}} \neq S'_{\text{new}}$ fails then a security exception will be raised and the grouped items could be moved to a secure and controlled area where each one could be further investigated. Our proposal has the added advantage that it can authenticate any subset of the group or even individual tags using exactly the same protocol steps. For example, if the protocol is only run with one tag the verifier checks if $s_{i_{\text{new}}} = f_2(s_{i_{\text{current}}})$ to determine whether the tag is legitimate. In this case the two group states are conventional bit strings with no collisions.

IV. Security Analysis

The group authentication protocol is meant to prove that a group is complete and pure. The condition for the group being complete is as follows:

- the verifier must observe cn bit collisions in the tags' authentication states S_{old} and S_{new}, where n is the number of tags in the group and c is the number of collisions attributed to each tag.

If less than cn collisions are observed it therefore means that at least one tag must be missing from the group. An attacker wishing to remove an item, and by implication a tag, from the group might substitute legitimate tags with devices, which attempt to replicate the responses of the legitimate tags. For a group to be pure the following conditions must hold:

- the group must be complete.
- the bit positions of all bit collisions in S_{new} must be correct, i.e. these must occur in the expected bit periods.
- the bit values received during the bit periods where no bit collisions occur must be correct.

If an attacker tries to replace a stolen tag it must ensure that his fake tag's bit responses still result in a valid S_{new}. In other words, without knowing the group key k_g the attacker must cause bit collisions in the same bit positions as the tag it replaced. The attacker must also guess the values of the remaining non-collision bits otherwise his substitute device would cause additional bit collisions in bit positions where none are expected.

We consider five attack cases where an attacker attempts to remove an item from a group of three or more items. In each case we state the probability p_{a} that the attack will not be detected. We assume that the attacker knows the value of the current group authentication state S_{current}, i.e. it observed S_{new} during the previous protocol run, but that it has no knowledge of the group key k_g.

Case 1 – Attacker uses a simple tag: The attacker's replacement tag functions as a normal tag, i.e. it adheres to the protocol rules and does not know what the other tags are transmitting. Although the attacker knows S_{current} this does not assist him a great deal in calculating S_{new} as it does not know which bit collisions are contributed by the tag it replaced, which bit values need to be transmitted to cause the bit collisions or what the updated non-collision values will be. The attacker would need to contribute bit collisions in the correct bit positions and transmit the correct non-collision values in order to not introduce extra collisions. To succeed the attacker essentially needs to guess the right bit values in all positions where other tags do not cause bit collisions. The probability of an attack succeeding if the attacker removes $n_{\text{a}} \leq n - 2$ tags from a group of n tags, each contributing c collisions, can be calculated as follows:

$$p_{\text{a}} = \left(\frac{1}{2}\right)^{c(n+n_{\text{a}})} \tag{1}$$

If the attacker replaces $n_{\mathrm{a}} = n - 1$ tags the attack probability is given by

$$p_{\mathrm{a}} = \left(\frac{1}{2}\right)^{2cn} \tag{2}$$

because the single legitimate tag left cannot contribute any collisions by itself and therefore the attacker would need to guess all the bit values correctly. We only consider the case where the attacker removes $n_{\mathrm{a}} \leq n-2$ tags for the remaining cases.

Case 2 – Attacker uses a quiet tag: The attacker's replacement tag does not strictly adhere to the protocol but it does not know what the other tags are transmitting. In this case the attacker's strategy is to only transmit bit values in the positions where it thinks the tag would need to cause a bit collision. The attacker's tag stays quiet the rest of the time. The main benefit to the attacker is that it does not need to guess the correct values of the non-collision bits, as these are still transmitted by the other tags, and does therefore not risk causing extra bit collisions. To succeed the attacker needs to select the correct bit positions in which to cause collisions and choose the right bit value that will result in a collision, i.e. the attacker needs to transmit a different symbol as the rest of the tags. The attacker first needs to guess which of the bit pairs contain a collision caused by the tag it replaced. In the case where the attacker removes $n_{\mathrm{a}} \leq n-2$ tags from a group of n tags that each contribute c collisions the probability of this attack succeeding can be calculated as follows:

$$p_{\mathrm{a}} = \begin{pmatrix} cn \\ cn_{\mathrm{a}} \end{pmatrix}^{-1} \cdot \left(\frac{1}{2}\right)^{2cn_{\mathrm{a}}} \tag{3}$$

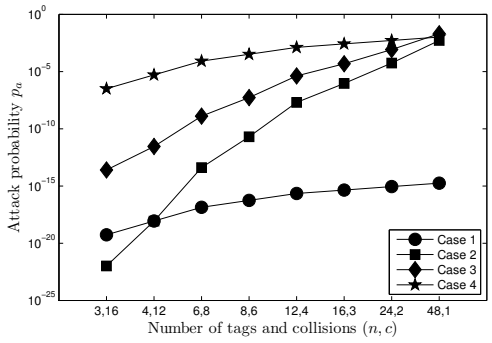

Fig. 3. Single tag removed if the length of S and s_i is 96(a realistic response length for both EPC (ISO 18000-6C) and ISO 15693/18000-3 tokens)

Case 3 – Attacker uses a smart tag: The attacker's replacement tag does not adhere to the protocol and can observe what the other tags are transmitting. We assume that the tag can determine the bit value and prepare its response right at the start of the bit period, even thought this might be unrealistic in practice. The primary benefit is that if the attacker wants to contribute a bit collision it knows the values the other tags are transmitting and therefore it simply needs to transmit the alternative value. The attacker can also observe bit collisions, which could help it to choose the bit positions in which it causes bit collisions. Obviously, the attacker would not need to guess the values of the non-collision bits as it could either choose not to transmit anything or simply learn the correct value from the other tags. For a group with n tags, which each contribute c collisions, the probability of this attack succeeding if $n_{\mathrm{a}} \leq n-2$ tags are replaced can be approximated by:

$$p_{\mathrm{a}} = \begin{pmatrix} (cn + cn_{\mathrm{a}})/2 \\ cn_{\mathrm{a}} \end{pmatrix}^{-1} \cdot \left(\frac{1}{2}\right)^{cn_{\mathrm{a}}} \tag{4}$$

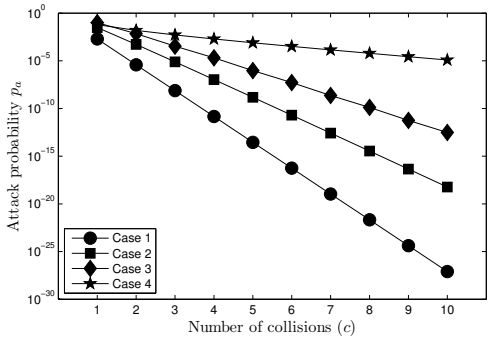

Fig. 4. Effect if the number of collisions per tag increases and group size is constant($n = 8$)

Case 4 – Attacker knows tag state and collision positions: In this case the attacker knows the current authentication states of the tag it wants to replace and at least two other tags. This allows the attacker to identify which bit collisions are contributed by the tag it replaces by comparing its state with the other tags' states and seeing what bit values differ, i.e. if the other two tags have the same bit value and the replaced tag's bit value differs it is contributing a collision at that bit position. This is truly a worst case scenario as this is unlikely to happen during normal operation of the system. The attacker can only eavesdrop the group state S, which as a whole does not reveal much information about individual tag authentication states. As a result, an attacker would need to be in a position to observe $s_{i_{\mathrm{new}}}$ in protocol runs between the verifying node and individual tags. For the simple tag scenario an attacker still needs to guess the bit pair rotation in addition to the non-collision values and the bit values needed to cause collisions as these are randomly changed by the XOR operation. The attacker's best approach is still to randomly guess the tags' individual authentication states and therefore the probability that the attack succeeds is still represented by Equation 1. In the quiet and smart tag scenarios the attack success probability increases. Since the attacker knows the bit pairs in which his tag contributes bit collisions and it only has to guess how many bit pairs the authentication state is going to rotate and whether the bit pair values will swap to know in which bit positions the collisions should be contributed in the updated group authentication state. For the general smart tag case, with $n_{\mathrm{a}} \leq n-2$, the probability of the attack succeeding becomes

$$p_{\mathrm{a}} = \frac{1}{cn} \cdot \left(\frac{1}{2}\right)^{cn_{\mathrm{a}}} \tag{5}$$

For the quiet tag scenario the attacker also needs to guess the bit value that will cause the bit collision, so the new attack success probability can be written as follows:

$$p_{\mathrm{a}} = \frac{1}{cn} \cdot \left(\frac{1}{2}\right)^{2cn_{\mathrm{a}}} \tag{6}$$

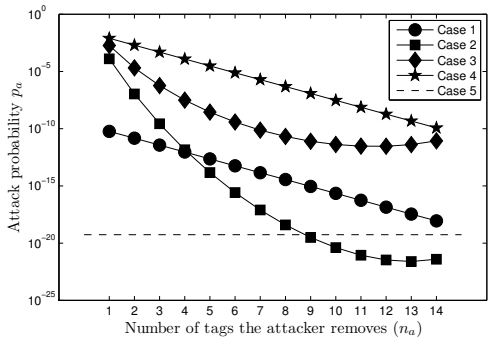

Fig. 5. Attacker replaces multiple tags in the same group ($n = 16, c = 2$)

Case 5 – Attacker creates a new group: The attacker attempts to create a whole new group to hide the fact that it took an item or multiple items. In this case it has to guess all the bit collision positions and the values of the non-collisions within the updated group authentication state S. The advantages the attacker gains in Case 2 and Case 3 no longer apply as there are no legitimate tags to observe or rely upon to transmit the correct non-collision bit values. This means that the attacker needs to guess the bit position of the collision and the value of the non-collision bit position in each bit pair. It therefore has a $\left(\frac{1}{2}\right)^2$ chance of guessing each bit pair correctly. For a group of n tags, with c collisions per tags, the probability that the attacker successfully creates a new group is therefore given by

$$p_{\mathrm{a}} = \left(\frac{1}{2}\right)^{2cn} \tag{7}$$

The effects of the group size, the collisions per tag and the attacker removing multiple tags on the probability that an attack succeeds for the different cases are illustrated in Figures 3, 4 and 5 respectively. The probability for Case 4 in each figure was calculated using Equation 5 as it represents the best case for the attacker.

A. Additional Security Comments

Privacy: The use of a fixed group ID does allow for illegitimate tracking of a group. However, in most cases privacy becomes a real concern after the user takes ownership. This protocol is meant for use during shipment, and provides reasonable item level obfuscation, i.e. Mr A is receiving a shipment from the chemist versus Mr X is receiving drug Y and X from the chemist, or Miss Y has received a shipment from the library rather than Miss B has received books M and N.

Group keys: Even thought not all RFID devices are currently tamper resistant, we assume that tags used in this scheme can reliably store a secret key. Retrieving the key off a tag compromises the tag's group, but does not effect other groups. This should be considered when deciding on group size and a large set of related tags might be divided in several groups.

Synchronization: It is not possible for an attacker to desynchronise the tags simply by interfering with one of the data exchanges. Once the authentication request received from the reader in step 3 is verified, the tags will update to a new authentication state. The new state will be sent to the reader in the next protocol run and the protocol will essentially self-synchronise. To advance the sequence number seq the attacker would need to generate a valid m_1, which is not feasible without knowledge of the group key k_g. The attacker could run a partial protocol execution, up to step 3, to obtain a valid m_1 but the verfier woul subsequently raise a security exception if step 4 is not completed..

Relay Attacks: Relay attacks are very powerful, circumventing all application layer security protocols. An attacker can replace the entire group with a proxy device and relay communication between the verifier and the removed group. However, of the attacker only removes some tags the protocol is somewhat resistant to this attacker needs to provide the removed tags' response at the same time as the tags still present in the group. In an ISO 14443 system this would mean that an attacker would need to relay data in less than 2.4 μs [21], which is faster than the 20 μs [22] practically needed in an optimised attack.

V. Comments on Practical Feasibility

The proposed scheme proposed in is intended to operate in a system where a single reader communicates with a group, because the reader has to receive the replies from all the tokens to identify the bit collisions. Some inventory management systems currently use such an architecture, especially when accurate read zone control is required to identify items within a small area, e.g. items moving past on a conveyor belt [23]. This architecture is likely to be used in applications that identify items within a specific container or package, i.e. in systems with items are grouped and could benefit from the proposed scheme.

The length of the tokens authentication state is limited by memory and transmitted message length constraints. The number of tokens that can be activated and read simultaneously by a single reader might also be limited by practical aspects, such as the receiver architecture of the reader, the operational range of the reader and the number of tokens that can be powered at the same time from the transmitted carrier. However, this scheme can stil offer benefits even if the group size is limited. For example, a pharmaceutical system could be monitoring containers containing individually tagged blister packs. A system that needs to process 120 containers a minute, each containing 15 blister packs, would normally perform 1800 authentication operations a minute and require a reader and tokens that could execute the required cryptographic functions in approximately 33 ms. Using the proposed scheme the

system only has to perform 120 authentication operations a minute and the reader and the token has 500 ms to execute the required functions. Alternatively, the system throughput could be increased to monitor 1800 containers every minute using the same reader and tokens as before.

Although UHF technology is often associated with inventory management there are also numerous such systems utilising HF technology [24]. In HF systems adhering to the ISO 14443 [13] and ISO 15693 [14] (ISO 18000-3 [15]) standards bit collisions are used during tag selection and the receiver architectures of existing readers also allow for the detection of such collisions. Our scheme requires a tag that could implement pseudo-random function f_1 and the basic bit permutation function f_2. There are a number of commercial tokens adhering to ISO 14443 and 15693 that currently implement cryptographic functions of similar or greater complexity, e.g. NXP Mifare (ISO 14443) and HiTag (ISO 15693) products. Although commercial UHF tags with such capabilities do no exist at the moment, there are examples of experimental passive tags at this frequency that does have the capability to implement this protocol [25]. It is therefore feasible that the proposed scheme could be deployed in inventory management systems using existing technology.

VI. Conclusion

In this paper we present a novel group authentication protocol that combines application layer cryptographic mechanisms with physical aspects of the communication channel to verify whether a group of RFID tags is complete and pure. Multiple tags can transmit their authentication responses simultaneously as the verifier uses the resultant controlled bit-collision pattern, and not the individual responses, to authenticate the tags. Thus, the verifier need not authenticate each tag sequentially and it can therefore authenticate multiple tags within a time period comparable to the time taken to perform a single challenge-response sequence. This significantly reduces the transaction time when processing a high volume of items at any specific reader, which is a realistic problem in RFID systems where the available transaction time is limited as the result of high tag throughput. The protocol uses only a keyed pseudo-random function and simple bit permutation operations, which is comparable to the cryptographic primitives required by most grouping proofs and lightweight authentication protocols proposed for the RFID environment. Bit collisions are already used in anti-collision methods in current systems and some commercial tokens currently implement cryptographic functions of similar or greater complexity than those required in this proposal, making it feasible that the proposed scheme could be implemented using existing technology. Future work would investigate the addition of group privacy, the possibility of new or existing UHF PHY/MAC layers supporting this scheme, practical implementation on HF and UHF platforms and experiments to determine the group size that would work reliably in current off-the-shelf systems.

References

[1] G. Avoine, "Bibliography on security and privacy in RFID systems," Web source, 2010, http://www.avoine.net/rfid/.

[2] A. Juels, "RFID security and privacy: A research survey," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 381–394, Feb. 2006.

[3] M. Langheinrich and R. Marti, "Practical minimalist cryptography for RFID privacy," *IEEE Systems Journal*, vol. 1, no. 2, pp. 115–128, Dec. 2007.

[4] A. Juels, "Yoking-proofs for RFID tags," in *Workshop on Pervasive Computing and Communication Security*. IEEE, Mar. 2004, pp. 138–143.

[5] J. Saito and K. Sakurai, "Grouping proof for RFID tags," in *Advanced Information Networking and Applications*. IEEE, Mar. 2005, pp. 621–624.

[6] S. Piramuthu, "On existence proofs for multiple RFID tags," in *Conference on Pervasive Services*. IEEE, Jun. 2006, pp. 317–320.

[7] L. Bolotnyy and G. Robins, "Generalized yoking-proofs for a group of RFID tags," in *Conference on Mobile and Ubiquitous Systems*. IEEE, Jul. 2006, pp. 1–4.

[8] C. Lin, Y. Lai, J. D. Tygar, C. Yang, and C. Chiang, "Coexistence proof using chain of timestamps for multiple RFID tags," in *Advances in Web and Network Technologies, and Information Management*. Springer LNCS 4537, Aug. 2007, pp. 634–643.

[9] Y. Lien, X. Leng, K. Mayes, and J. Chiu, "Reading order independent grouping proof for RFID tags," in *Intelligence and Security Informatics*. IEEE, Jun. 2008, pp. 128–136.

[10] M. Burmester, B. de Medeiros, and R. Motta, "Provably secure grouping-proofs for RFID tags," in *Smart Card Research and Advanced Applications*. Springer LNCS 5189, Aug. 2008, pp. 176–190.

[11] W. Luo, S. Chen, T. Li, and Y. Qiao, "Probabilistic missing-tag detection and energy-time tradeoff in large-scale rfid systems," in *ACM Symposium on Mobile As Hoc Networking and Computing*. ACM, 2012.

[12] J. Mitsugi, Nakamura, and J. Murai, "Theory and performance evaluation of group coding of rfid tags," *IEEE Transactions on Autmation Science and Engineering*, vol. 9, no. 3, pp. 458–466, Jul. 2012.

[13] ISO/IEC 14443, "Identification cards – Contactless integrated circuit cards – Proximity cards," Technical Standard, 2001.

[14] ISO/IEC 15693, "Identification cards – Contactless integrated circuit cards – Vicinity cards," Technical Standard, 2001.

[15] ISO/IEC 18000, "Information Technology AIDC Techniques-RFID for Item Management – Air Interface," Technical Standard, 2008.

[16] C. Castelluccia and G. Avoine, "Noisy tags: Pretty good key exchange protocol for RFID tags," in *Smart Card Research and Advanced Applications*. Springer-Verlag LNCS 3928, Apr. 2006, pp. 289–299.

[17] E. Haselsteiner and K. Breitfuss, "Security in near field communication," in *Workshop on RFID Security*, Jul. 2006, pp. 3–13.

[18] M. Rieback, G. Gaydadjiev, B. Crispo, R. Hofman, and A. Tannenbaum, "A platform for RFID security and privacy administration," in *Large Installation System Administration Conference*. Usenix, Dec. 2006, pp. 89–102.

[19] A. Juels, R. Rivest, and M. Szydlo, "The blocker tag: Selective blocking of RFID tags for consumer privacy," in *Computer and Communications Security*. ACM, Oct. 2003, pp. 103–111.

[20] G. Hancke, "Modulating a noisy carrier signal for eavesdropping-resistant HF RFID," *Springer e & i Elektrotechnik und Informationstechnik*, vol. 124, no. 11, pp. 404–408, Nov. 2007.

[21] G. Hancke and M. Kuhn, "Attacks on time-of-flight distance bounding channels," in *Conference on Wireless Network Security (WISEC 2008)*. ACM, Mar. 2008, pp. 194–202.

[22] K. M. G.P. Hancke and K. Markantonakis, "Confidence in smart token proximity: relay attacks revisited," *Computers & Security*, vol. 28, no. 7, pp. 615–627, Oct. 2009.

[23] Impinj, Inc, "RFID case study: Purdue Pharma," Web source, jan 2009, http://www.impinj.com/applications/case-studies/pharmaceutical-industry.aspx.

[24] Aberdeen Group, "Where RFID meets ROI: Beyond Supply Chains," Web source, nov 2008, http://www.aberdeen.com/Aberdeen-Library/5296/RA-radio-frequency-identification.aspx.

[25] A. Sample, D. Yeager, P. Powledge, A. Mamishev, and J. Smith, "Design of an RFID-based battery-free programmable sensing platform," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 11, pp. 2608–2615, Nov. 2008.