An Automatic, Time-Based, Secure Pairing Protocol for Passive RFID

***Abstract*—This paper introduces the Adopted-Pet (AP) protocol, an automatic (i.e. requiring no human interaction) secure pairing protocol, in which two pairing devices start trusting each-other after spending long intervals of uninterrupted time in the proximity of each other. We show how the protocol can be secured against occasional malicious entities, and we provide a possible implementation. Our proposed implementation of the AP protocol is based on a simple LFSR (with the alternative of using a shrinking generator), which makes it ideally suited for low-cost passive RFID tags.**

I. Introduction

Recent technological advances, combined with an increasing demand for automatic inventory and tracking, provide a glimpse of a near future where radio-frequency identification (RFID) becomes ubiquitous. From industrial platforms to home environments, from retail stores or restaurants to medical facilities, the potential use of passive RFID tags is only limited by human imagination.

Unfortunately, as with any relatively new technology, numerous controversies still surround the wide-scale deployment of RFID. In fact, consumers' fear of privacy invasion already triggered several small-scale street protests around the world. Indeed, current RFID tags are subject to privacy attacks, which could make it feasible to track the movements of a person (the same idea is non-malevolently used to track patients in clinics or senior citizens in assisted-living facilities), or to inventory their personal possessions.

As an immediate countermeasure, pairing algorithms could be implemented to ensure that the RFID tags only respond to very few authenticated tag readers. Although device pairing protocols abound in the literature [], very few of these protocols can be implemented into the cost-constrained, resource-limited passive RFID tags. Moreover, most low-complexity pairing protocols require human interaction to complete the pairing process. Human interaction is generally not desirable because it is often viewed as an additional burden on the consumer, who may disregard proper protocol, leading to faulty pairings or omissions.

For example, the *Resurrecting Duckling Protocol* [] requires that the new tag should only trust the first reader it "sees" when it is resurrected. In order to transfer ownership of the tag, the trusted reader has to "kill it" (only the trusted reader can perform this action). The Resurrecting Duckling has several drawbacks. First, both killing the duckling and resurrecting it require human interaction. Second, if the duckling is accidentally "killed" by a malicious user, the legitimate user looses access to the tag.

In this paper, we are proposing an automatic, time-based pairing protocol, which we denote the *Adopted-Pet (AP) Protocol*. As opposed to the *Resurrecting Duckling protocol*, our *Adopted Pet Protocol* provides a more natural way of secure association between the tag and the reader in the home environment. Just like when adopting a new pet from the animal shelter, in our protocol trust is earned by spending time together. Once brought into the home environment, the new tag will start being *courted* by a home reader. After the tag and the reader spend a pre-programmed amount of time together (usually over night), the tag starts trusting the reader, and responding to its queries.

Note that the AP protocol requires no human interaction. Moreover, if the tag accidentally begins to trust another reader (for example during a trip), the home reader can re-gain the tag's trust upon return. Our time-based trust-earning AP protocol raises several serious challenges. To better understand them, consider the two real-life scenarios, which are crucial in defining the properties of our AP protocol.

Scenario 1 – legitimate pairing: A new tag is brought into the home environment. The home readers begin courting the tag. They need to gather enough information about the tag, so that they will be able to prove to the tag that they have been around for a period of time which defines them as *legitimate readers*.

Scenario 2 – the man on the bus: A certain tag is carried around by its owner, and subjected to the owner's daily routine. The routine includes a several-hour bus ride to work. The attacker happens to also ride the same bus, and his customized reader attempts to pair with the tag. However, the attacker cannot spend more than several hours a day in the proximity of the tag, without being detected.

It is important to note that passive RFID tags generally have no internal time reference. A tag's only notion of time comes from successive reader queries. This makes it unfeasible for the tag to keep track of the actual time spent in the company of a certain reader per day. If in Scenario 1 the consumer takes the tag outside the home for a short while – and while outside the home, the tag is be interrogated by other readers – (or if the tag becomes inaccessible to the reader due to some form of temporary interference), then upon return, the tag does not know whether it's been a minute or a day. On the other hand, if the tag would just count the number of queries from a certain reader, and pair with the reader after a certain threshold is reached, then it would only take several days for the "man on the bus" in Scenario 2 to gain control of the tag.

In conclusion, to differentiate between the legitimate reader and the "man on the bus", it does not suffice to consider only the quantity of time spent in the presence of the tag. Rather, we need to take advantage of the time quality: we can expect that the time that the legitimate reader spends with the tag is less likely to exhibit large interruptions.

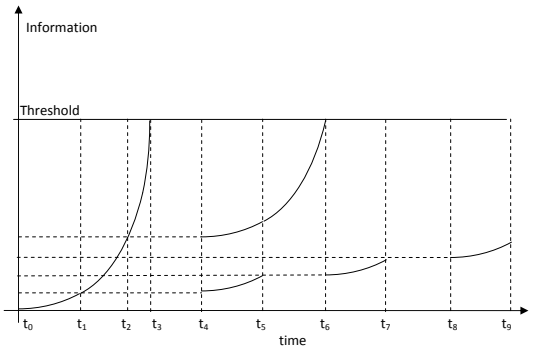

Fig. 1. Gathering information about the secret key for the AP protocol, in an exponential-leakage-rate system.

II. The Adopted-Pet (AP) Protocol

Our Adopted-Pet (AP) protocol relies on the fact that a tag-reader located inside the tag owner's home should spend more *continuous* time in the presence of the tag than any other reader located anywhere outside the home. By *continuous* time we mean an interval of time during which the tag is not interrogated by any other reader. If the tag is interrogated by a different reader between two such continuous time intervals, we say that the tag and the initial reader have become *desynchronized*.

Naturally, a reader located in the tag owner's place of work might have an advantage similar to a reader located in the home environment, but we should assume that the place of work is generally a safe environment (otherwise, the tag owner has more serious problems than his RFID tag being inventoried).

The tag will only respond with its Universal Product Code (UPC) if the reader querying it proves that it knows the tag's secret password. Otherwise, the tag will assume that the reader is not legitimate, and it will respond with a piece of information related to the secret password – throughout this paper we shall call this "*a clue*". If the reader keeps querying the tag, it will eventually accumulate enough information to learn the tag's secret password. However, if a certain malicious reader queries the tag for many short continuous time intervals, such that between any two such time intervals the tag receives an unknown number of queries from other readers, the information extracted from the tag's responses should be of little value to the malicious reader.

We envision a system where a reader can gather information about the tag's secret password at a rate that starts at an initial value, and increases exponentially with every new tag response. However, if the reader and the tag become desynchronized, the rate of gathering information returns to its initial value, and starts increasing from there. Note that this does not imply that the information obtained by the reader during the first continuous time interval is lost. When the gathered information reaches a certain threshold, the tag's secret key can be learned. We shall refer to such an idealized system as an "*exponential-leakage-rate system*". The gathering of information is represented in Figure 1, for three different scenarios: in a single continuous time interval ($[t_0, t_3]$), in two continuous time intervals ($[t_0, t_2]$ and $[t_4, t_6]$), and in many short continuous time intervals ($[t_0, t_1]$, $[t_4, t_5]$, $[t_6, t_7]$, $[t_8, t_9]$, ...).

In the spirit of recycling, we propose to use cryptosystems which loose information about their secret key at such an increasing rate, but only if surveyed continuously. As an intuitive (although neither practical, nor desirable) example, the password could be the title of a book from a library. The tag could transmit one letter from the book, in order, every minute. If the reader listens continuously for a large period of time, it will obtain an entire chapter, and probably be able to identify the book. Similarly, if the reader listens for distinct continuous time intervals, it will obtain substantial pieces of two different chapters, probably leading to the identification of the book as well. However, if an attacker can only listen over many really short continuous-time intervals, it will get a meaningless set of words, without even a position reference in the book. Clearly, this example does not have good security properties, since an attacker might identify the book by focusing on very peculiar words, or by looking at word frequencies.

For a more practical solution, consider the well-known linear-feedback shift register (LFSR). Assume that the tag contains an internal LFSR, of length L, the characteristic polynomial of which (of order L) is programmable by an authorized reader, and constitutes the tag's secret password. If the reader proves to the tag that it knows the characteristic polynomial of its LFSR, the reader is considered authorized, and the tag responds to its queries with the UPC, and allows the reader to re-program the LFSR.

On the other hand, if the RFID tag does not trust the querying reader, it generates a single bit with its internal LFSR (of length L), and responds with this bit. The reader memorizes the bit. If the reader can gather $2L$ contiguous bits, it can then solve for the coefficients of the LFSR's characteristic polynomial – a linear system of L equations with L unknowns. Note that efficient recursive methods for solving the system may be implemented in the reader, such as the Berlekamp-Massey algorithm[].

As an example, the RFID tag can be throttled [] to respond only once every minute. If the tag sports an internal LFSR of length $L = 300$, any reader that spends a continuous time of 10 hours in the tag's proximity can figure out the characteristic polynomial. Similarly, if the reader can only afford to spend 7.5 hours a day with the tag, the characteristic polynomial can be discovered after only two days. On the other hand, if an attacker can only afford to spend less than 5 hours a day close to the tag (this would include any possible "man on the bus" type of attacker), he can only gain access to a contiguous bit sequence of length $L = 300$. However, this leaks absolutely no information about the characteristic polynomial, since any LFSR of length at least 300 is capable of producing any non-zero 300-bit sequence. Even if the attacker gathers a large number of such L-bit sequences, over the course of a year, the *ubiquitous RFID environment* ensures that the number of bits outputted by the tag's internal LFSR between two attacker sessions is unknown and unpredictable. Hence, the attacker's information is completely useless.

In the following sections we provide a more complete

analysis, including an alternative implementation, and how to deal with small tag-reader desynchronizations.

III. Design Considerations

Since this paper is concerned with the application of the AP protocol to passive RFID pairing, and passive RFID tags are characterized by small computational power and memory, we propose that the protocol implementation use binary, linear finite state machines (FSMs).

Let $z_0, z_1, z_2, \ldots$ be a sequence of bits generated by one such binary, linear finite state machine. Such a sequence is eventually periodic, because of the finiteness of the state space. Consequently, it is possible to generate the sequence via a linear recurrence relation like

$$z_{j+L+1} = c_0 z_j + c_1 z_{j+1} + \cdots + c_{L-1} z_{j+L} \tag{1}$$

with $c_0, c_1, \ldots, c_{L-1} \in \{0, 1\}$. Computations are performed in the two-element field, $\mathbb{F}_2$. This is nothing but an algebraic formulation of a *linear feedback shift register*. Note that the entire sequence is determined by the coefficients $c_0, \ldots c_{L-1}$ and the initial seed $z_0, \ldots, z_{L-1}$.

The smallest value of L for which a recurrence such as (1) exists is called the *linear complexity* of the sequence. The linear complexity of an actual LFSR implemented in hardware is, of course, equal to the register length. However, when a large linear complexity is required, it may become unpractical to use a simple LFSR. Take, for instance, our example above, and assume that the tag is required to respond to interrogations at least once every second (instead of once every minute). To maintain the same characteristics, the tag's internal LFSR should have length $L = 18000$. But the implementation of such a long LFSR would almost exhaust all the resources available for commonly-used passive tags.

A viable alternative is to use stream cipher designs with irregular clocking, for which the linear complexity is exponential in the number of registers.

For the linear, binary FSMs discussed above, suppose that a reader obtains a sequence of contiguous bits $z_0, z_1, z_2, \ldots$ created according to the recurrence (1) and wishes to determine the coefficients $c_0, c_1, \ldots, c_{L-1}$. The sequence of bits satisfies the following system of linear equations:

$$\begin{aligned}
c_0 z_0 + c_1 z_1 + \cdots + c_{L-1} z_{L-1} &= z_L \\
c_0 z_1 + c_1 z_2 + \cdots + c_{L-1} z_L &= z_{L+1} \\
c_0 z_2 + c_1 z_3 + \cdots + c_{L-1} z_{L+1} &= z_{L+2} \\
&\vdots \\
c_0 z_{L-1} + c_1 z_L + \cdots + c_{L-1} z_{2L-2} &= z_{2L-1},
\end{aligned} \tag{2}$$

in the unknowns $c_0, \ldots, c_{L-1}$. Such a system is easily solved for moderately-large linear complexities L. For example, using ordinary Gaussian elimination, the system is solvable in time $O(L^3)$. Note here that the linear complexity of the FSM is assumed known in advance. Nevertheless, even if the linear complexity is unknown, one could simply try every possible linear complexity, starting from 1. Using Gaussian elimination, the solution (and the linear complexity) would be found in time at most

$$O(1^3) + O(2^3) + \cdots + O(L^3) \leq L \cdot O(L^3) = O(L^4).$$

For very large linear complexities L, a reader could implement the Berlekamp-Massey algorithm [3] which, for every new received bit, computes the smallest-size LFSR that could have generated the currently-available sequence of bits. Moreover, the Berlekamp-Massey algorithm does not require that the linear complexity be known in advance.

IV. Dealing with Reader-Tag Desynchronizations

To formulate and solve the system in (2), we implicitly made the following two assumptions: (1) $2L$ bits of the sequence are known, and (2) the known bits are consecutive. The first assumption is essential. It is not hard to find two distinct sets of coefficients which generate sequences that coincide on $2L - 1$ consecutive bits. We emphasize that in this discussion we are treating the sequence $z_0, z_1, \ldots$ as being generated by a black box. If we assume some knowledge of the internals of the generation procedure, there may well be more efficient methods of analyzing the sequence.

Now we turn to the second assumption: the consecutiveness of the known bits. Suppose that instead of consecutive bits, we have obtained $2L$ bits, with k unknown bits interspersed among them. We wish to address the complexity of determining the unknown coefficients $c_0, \ldots, c_{L-1}$ in this scenario. By treating each missing bit as an additional unknown, one can still set up a system of equations as in (2), but now there will be $L + k$ equations in $L + k$ unknowns. However, the system is no longer linear: it becomes quadratic. This is because if z_j is an unknown representing a missing bit, then the system will contain quadratic terms of the form $c_i z_j$, for $i = 0, \ldots, L-1$. In fact, of the $(L + k)^2$ terms in the system, approximately Lk of them will be quadratic.

Unlike the situation for linear systems, there is no known efficient method of solving a quadratic system over a finite field. The problem of finding one solution to a quadratic system over our field $\mathbb{F}_2$ is NP-complete [2], and known in the literature as the "MQ problem." Note that NP-completeness is a worst-case analysis, and it is entirely possible that the system that would arise in this particular application has a polynomial-time solution. However, none of the algorithms with which we are familiar seem to apply here. In particular, algorithms such as XSL [1] are not appropriate since the system is not sparse. Furthermore, the quadratic system may not have a unique solution. It may require additional equations (and therefore, additional known bits) to obtain uniqueness.

Of course, one could always find the solution through exhaustive search. Note that for each of the 2^k choices for the unknown z-bits, we are reduced to a linear system (in the unknown c_i's) that we can solve by Gaussian elimination or by the Berlekamp-Massey algorithm []. Alternatively, we can try each of the 2^L choices for the c_i's and solve the resulting linear system for the unknown bits. The running time for this approach is therefore $O(2^{\min(n,k)})$.

While this running time ensures that an exhaustive-search attack is not feasible for the man on the bus, the exhaustive-search method may be a good solution for approaching small desynchronizations between the tag and the legitimate reader (as stated in Section II above, such desynchronizations can appear as a result of the user taking the tag outside for a small time interval, or of temporary interference, such as a cell phone positioned close to the tag).

If, for example, the tag responds to each query with a newly-generated bit, accompanied by the previous three or four bits, the reader can easily (and with good probability) identify such desynchronizations, and, as soon as it has collected enough information, proceed with an exhaustive search for the missing bits.

V. The Dangers of Routine

One of our main assumptions at the beginning of this paper was that our protocol will function in a *ubiquitous RFID environment*. This assumption ensures that throughout the day the tag is being interrogated numerous times, by various readers. When we stated that "the number of bits outputted by the tag's internal LFSR between two attacker sessions is unknown and unpredictable", we have implicitly assumed that the number of interrogations from "non-legitimate" readers (which force the tag to run its internal finite state machine and transmit a clue) is a random variable with a somewhat uniform distribution.

However, in reality, the owner of a tag is expected to follow a daily routine. Moreover, even with a ubiquitous RFID environment, we can expect that the number of mobile, active RFID readers carried by other people is quite limited. This means that the majority of readers which interact with the tag are stationary (probably mounted in public places, such as a train station or a bar), and thus a certain daily routine could result in roughly the same daily number of tag interrogations by non-legitimate readers.

We model this phenomenon by assuming that the tag is interrogated daily by exactly the same number of stationary readers, and in addition, by a varying number of mobile readers. We denote the number of interrogations from stationary readers by R_s, and we model the number of interrogations from mobile readers by a discrete random variable M with a Poisson distribution of mean R_m. The attacker has access to a number n of contiguous bit sequences of length $S < L$ extracted from the output of the tag's linear binary finite-state machine, but separated by $R_s + M$ bits. A serious attacker, who follows the tag's owner from a distance, could recreate his daily routine, and thus learn the values of both R_s and R_m. When the value of R_m is relatively small, the attacker can assume that $R_s + M$=$R_s + R_m$, and attempt to correct for the error in this assumption by using an exhaustive-search method of the type described in the previous section.

The attacker is now in the position where he needs to find the characteristic polynomial of the tag's internal LFSR (or equivalent LFSR, if the tag uses a different kind of linear FSM), given n sequences of contiguous bits, of length S, and separated by $R_s + R_m$ bits. However, due to the fact that even small-linear-complexity FSMs display very large periods of the output sequence (for linear complexity L, if the characteristic polynomial of the equivalent LFSR is primitive, the period is $T = 2^L - 1$), it is safe to assume that all the attacker's n sequences of S bits belong to the same LFSR period. Therefore, this "routine attack" is reduced to the large-desynchronization problem of Section IV.

References

[1] Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in cryptology—ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Comput. Sci.*, pages 267–287. Springer, Berlin, 2002. Updated version: http://eprint.iacr.org/2002/044.

[2] A. S. Fraenkel and Y. Yesha. Complexity of problems in games, graphs and algebraic equations. *Discrete Applied Mathematics*, 1(1-2):15 – 30, 1979.

[3] J. L. Massey. Shift-register synthesis and bch decoding. *IEEE Trans. Information Theory*, 15:122–127, 1969.