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

G. T. Amariucai [†], C. Bergman [*] and Y. Guan [†]

[†] Dept. of Electrical and Comp. Engineering, Iowa State University, gamari@iastate.edu, guan@iastate.edu
[*] Dept. of Mathematics, Iowa State University, cbergman@iastate.edu

Abstract. This paper introduces the Adopted-Pet (AP) protocol, an automatic (i.e. requiring no human interaction) secure pairing protocol, adequate for the pairing between a passive RFID tag and a reader. Most pairing protocols rely for their security on a certain advantage that the legitimate devices have over any malicious users. Such advantages include proximity (employing near-field communication) or secret keys that are either produced with the assistance of, or verified by, the legitimate user. The advantage exploited by our novel AP protocol is the amount of *uninterrupted time* spent by the two devices in the proximity (although not requiring near-field communication) of each-other. We discuss several implementation configurations, all based on pseudo-random bit generators, employing short-length LFSRs, and requiring no more than 2000 transistors. This makes the protocol ideally suited for low-cost passive RFID tags. For each configuration we show that the AP protocol is highly secure against occasional malicious entities.

Keywords: Automatic pairing protocol, time-based pairing, passive RFID.

1 Introduction

Recent technological advances, combined with an increasing demand for automatic inventory and tracking, provide a glimpse of a near future in which 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.

Passive RFID tags are small, barely-visible, extremely cheap and hence extremely resource-limited electronic devices, which can communicate wirelessly to a more powerful device—a "reader"—usually during an automatic inventory. The typical passive RFID tag has no internal power source, and runs its internal circuitry by harvesting power from the electromagnetic waves produced by the reader during a query. Rather than producing and transmitting an electromagnetic wave of its own, the tag responds to queries from the reader by *backscattering* [1], i.e. by modulating the waveform reflected back to the reader—this can be done by varying the load impedance of the tag's antenna.

Unfortunately, as with any relatively new technology, numerous controversies still surround the wide-scale deployment of RFID. In fact, consumers' fear of privacy invasion have 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 [2–5], 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. For example, the user is required to shake the two devices simultaneously in [3], or to push a button in [4, 5]. The *Resurrecting Duckling Protocol* of [2], requires that the new tag should be "killed" by its previous owner and then "resurrected" by the new owner. The "duckling" will then trust only the the first reader that attempts to communicate with it during resurrection. The immediate drawback is that if the duckling is accidentally "killed" and then "resurrected" by a malicious user, the legitimate user looses access to the tag.

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. In this paper, we are concerned with automatic pairing that would provide commercial-level privacy for recently-introduced applications like smart refrigerators [6, 7], or smart wardrobes or bookshelves, which periodically inventory their contents, and are able to provide suggestions like a shopping list, a matched outfit, or the location of a book. We should note that, while human interaction may be appropriate for the pairing of highly security-sensitive devices, such as personal computers, users should not be expected to perform dozens of check-in procedures after every trip to the grocery store, in order to pair each item with their smart refrigerator or pantry.

We address the pairing between a passive RFID tag and a reader by proposing an automatic, time-based pairing protocol, which we view as an alternative to the Resurrecting Duckling protocol of [2], and which we denote the *Adopted-Pet (AP) Protocol*. As opposed to the *Resurrecting Duckling protocol*, our *Adopted Pet Protocol* provides a more natural method of secure and *transient* [2] association between the tag and the reader in the home environment. Just as when adopting a new pet from the animal shelter, in our protocol trust is earned by spending a long, quality 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 overnight), the tag starts trusting the reader, and responding to its queries.

The main advantage of our AP protocol is that it 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.

Naturally, this has the drawback that, if the item to which the RFID tag is attached is stolen, the thief can use the tag with his own home readers. However, we believe this should be irrelevant, since the value of the RFID-enabled service is generally less than the intrinsic value of the object to which it is attached. In addition, any sensible thief would attempt to remove the RFID tags from the stolen products anyway, to destroy the evidence of the crime. Moreover, the security of most encrypted devices becomes questionable when the device is stolen, because the thief has an unlimited time for breaking the encryption.

Our contributions can be summarized as follows: (a) we introduce the novel idea of uninterrupted-time-based pairing, and define the *adopted-pet* (AP) pairing protocol; (b) we provide a robust implementation philosophy, which can tolerate interference and desynchronizations, and demands extremely limited resources; (c) we discuss four possible implementations of the protocol inside passive RFID tags, in detail, emphasizing their individual security features and resource requirements; (d) we provide an analysis of the reader's part of the protocol. The paper is organized as follows. Section 2 presents the system and threat models. The AP protocol is introduced in Section 3, along with the RFID-adequate implementation philosophy. Section 4 discusses the four practical design solutions, while Section 5 delves deeper into the protocol security. Finally, we provide some hardware-related considerations in Section 6, and some concluding remarks in Section 7. *Take-Away Points* are emphasized throughout the paper, to facilitate reading and information synthesis (nevertheless, the take-away points alone *do not* constitute a good summary of the paper).

2 System Model, Threat Model, and Challenges

We consider a ubiquitous RFID environment, in which RFID readers are mounted in most public places, such as grocery stores, bars or train stations, and are carried around by individuals, in the form of their smart phones or laptops. In our model, most products available in stores contain individual RFID tags, and cannot be individually re-programmed at checkout (for instance, to "kill" the resurrecting duckling in each product, the checkout reader would have to establish secure individual communication with each item – a simple inventory would not suffice). Moreover, we assume that most RFID tags are designed according to the same flexible RFID standard. The tags attached to various items need to be usable by the customers, inside their homes, with their smart refrigerators, bookshelves and wardrobes.

Nevertheless, the RFID tags should not allow themselves to be inventoried by illegitimate readers during their trip from the store to the customer's home, or during any subsequent times when the products to which they are attached are worn or carried around. Our threat model consists of illegitimate readers attempting to inventory the tag. Since commercial-grade RFID devices (readers and tags) are designed for close-range communication, we do not concern ourselves with illegitimate readers that might eavesdrop on the legitimate reader and impersonate the latter. This sort of attack is normally beyond the scope of pairing protocols. It is also beyond the scope of pairing protocols to deal with jamming attacks. For completeness, we should also mention that attacks where the malicious entity attempts to modify the information transmitted over the channel in a controlled fashion (as in [8]) are not practical for our protocol. This is due to the fact that, in our protocol, the only information-bearing signals are transmitted by the tag. To mount such an attack, an adversary would need to be in close proximity to the tag (to receive its query responses), achieve a certain level of synchronization with the tag (in the presence of any scattering phenomena), and transmit with power well above that of the tag's backscattered signal. Otherwise, since most modern RFID protocols use a variant of frequency-shift-keying modulation (F2F, FM0, MMS, etc.), based on detecting transitions rather than power levels, the results of such an attack would be totally unpredictable (hence categorized as *jamming*).

Our time-based trust-earning protocol raises several serious challenges. To better understand them, consider the following two real-life scenarios, which summarize the model for the environment in which our AP protocol is intended to function, and the model for the attacks that the AP protocol should be able to foil.

Scenario 1—legitimate pairing: A new tag is brought into the home environment. The tag is attached to a product, like a food or clothing item, and should be used by a home reader, such as one in a smart fridge or wardrobe [6, 7]. The moment the tag enters the radius of action of a reader, the reader begins "courting" the tag. The reader needs to gather enough information about the tag, so that it will be able to prove to the tag that it has been around for a long period of time, which defines it as a *legitimate reader.*

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.

Take-Away Point 1 *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 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 has 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 spent by the legitimate reader in the company of the tag is less likely to exhibit large interruptions than the time spent by the attacker in the proximity of the tag.

Take-Away Point 2 *To summarize, our pairing protocol should have the following characteristics:*

- *It should be automatic;*
- *It should enable the legitimate reader to pair with the tag after spending enough time in the presence of the tag;*
- *It should not allow the pairing between a tag and a reader based only on the short but many interactions between them;*
- *It should not rely on any absolute time references;*
- *It should be adequate for implementation in low-cost, resource-constrained passive RFID tags.*

3 The Adopted-Pet (AP) Protocol

Our Adopted-Pet (AP) protocol relies on the fact that a reader located inside the tag owner's home should be able to spend more *uninterrupted* time in the presence of the tag than any other reader located anywhere outside the home. By *uninterrupted* 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 uninterrupted 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 AP protocol is described in Figure 1. The reader attempts an inventory of all the tags within its reach. As soon as a tag accesses the medium, and the channel is proved to be clear (no collisions are detected) [1], the reader proves to the tag that it can be trusted. 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 (if the tag is new), the tag will assume that the reader is not legitimate, and it will respond with a *"no trust"* sequence, followed by a piece of information related to the secret password—throughout this paper we shall call this "*a clue*". If the reader keeps querying the new tag, for long uninterrupted time intervals, 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 uninterrupted 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. It is important to note that our protocol integrates naturally with the current RFID singulation protocols; that is, only minor changes need to

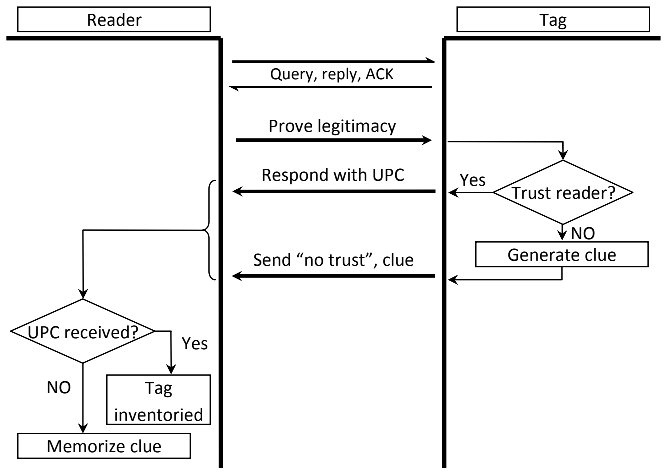

Fig. 1. An outline of the AP protocol.

be made to both reader and tag, while the physical and medium access control layers remain unchanged. Moreover, a single reader should be able to deal with large numbers of new (distrustful) tags simultaneously.

We envision a system in which 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 uninterrupted 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 2, for three different scenarios: (a) the information threshold is reached by a reader which queries the tag during the single uninterrupted time interval $[t_0, t_4]$, (b) the information threshold is reached by a reader that queries the tag during two distinct uninterrupted time intervals $[t_0, t_3]$ and $[t_5, t_7]$, and finally (c) an attacker queries the tag during the short uninterrupted time intervals $[t_0, t_2]$, $[t_5, t_6]$, and $[t_8, t_9]$, without reaching the information threshold. Note that our exponential-leakage-rate system also displays a time threshold $t_1 - t_0$. In order for a reader to obtain any information about the tag's secret password, the reader should query the tag for an uninterrupted time interval of at least $t_1 - t_0$. For example, no information is obtained by the attacker in the third scenario (c) during interval $[t_5, t_6]$.

In the spirit of recycling, we propose to use cryptosystems which lose information about their secret key at such an increasing rate, but only when 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

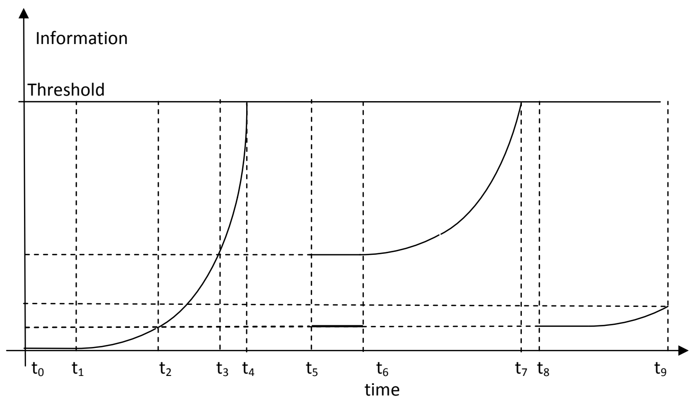

Fig. 2. Gathering information about the secret key for the AP protocol, in an exponential-leakage-rate system.

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 several medium-sized distinct uninterrupted 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 uninterrupted 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 following implementation philosophy, that relies on the well-known linear-feedback shift register (LFSR).

Take-Away Point 3 *Assume that the tag contains an internal LFSR, of length L, the characteristic polynomial of which (of degree 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 methods for solving the system may be implemented in the reader, such as the Berlekamp-Massey algorithm [9]. It is interesting to note that the legitimate reader is expected to* mount a successful linear-complexity attack *against the tag.*

The fact that an authorized reader is allowed to re-program the tag's LFSR provides a form of forward security – should the tag's secret leak at some point in time, the presence of the owner in a certain spot at a previous time should not be verifiable. Moreover, this ensures that any information about the tag's secret which leaks to an adversary (see Figure 2) becomes useless before the secret is breached.

Example 1 *Let us consider a concrete example. An RFID tag can be throttled [10] to respond only once every minute. If the tag has a built-in LFSR of length* $L = 300$*, any reader that spends an uninterrupted time of* 10 *hours in the tag's proximity can determine the characteristic polynomial. Suppose, instead, that the reader can only afford to spend* 7.5 *hours a day with the tag. The first equation can be written only after* 5 *hours of uninterrupted time (i.e.* $L + 1$ *queries), as discussed in more detail in Section 5.1. Hence, the characteristic polynomial can be discovered after only two days. On the other hand, if an attacker gains access to the tag for a single session of, say* 5 *hours (this would include any possible "man on the bus" type of attacker), he can only gain access to a contiguous bit sequence of length* $L = 300$*. This leaks absolutely no information about the characteristic polynomial, since any LFSR with a register 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 generated by the tag's internal LFSR between two attacker sessions is unknown and unpredictable. Hence, the attacker's information is completely useless both because the search space for the missing bits would be prohibitively large and also because there may be a great many different polynomials capable of producing each of the individual subsequences obtained.*

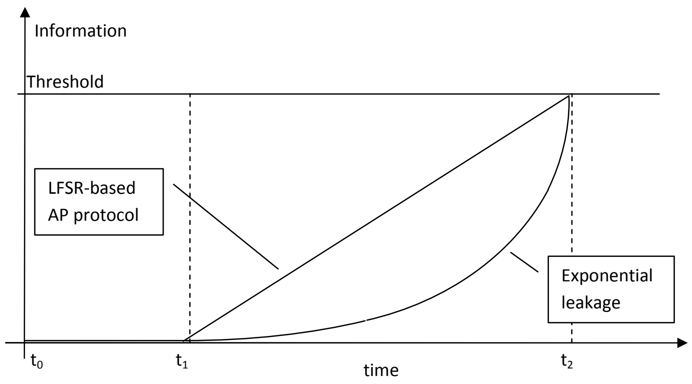

Fig. 3. The LFSR-based, vs. the exponential-leakage-rate system-based AP protocol.

Note that in Example 1 above, the number of possible configurations for the tag's internal LFSR is at least equal to the number of primitive polynomials of degree 301 over $\mathbf{GF}(2)$, which is given by $\frac{\phi(2^{301}-1)}{301} \geq \frac{\sqrt{2^{301}-1}}{301} \sim 10^{42}$, where $\phi(\cdot)$ is Euler's totient function.

Take-Away Point 4 *The LFSR-based AP protocol is a first-order approximation of our exponential-leakage-rate concept, as illustrated in Figure 3.*

Take-Away Point 5 *The AP protocol has all of the desired features of Take-Away Point 2:*

- *Since the tag responds to all querying readers (although it responds with the UPC only to trusted readers), giving untrusted readers a chance to earn their trust, the protocol is fully automatic;*
- *Since it is based on the legitimate reader spending large periods of uninterrupted time in the presence of the tag, a home reader (like a smart fridge or bookshelf) would be able to pair with the tag overnight;*
- *Since no information about the tag's secret is leaked until a relatively long interval of uninterrupted time is spent with the tag, the protocol is secure and guarantees user privacy;*
- *Since the tag is only concerned with verifying whether the reader knows its secret, and with running its internal pseudo-random bit generator, the protocol does not need the tag (or the reader) to keep an absolute time reference;*
- *Since our implementation philosophy relies on a simple LFSR, we should be able to implement the protocol with very little expense of resources.*

In the following sections we provide a more complete analysis, including alternative implementation, and how to deal with tag-reader desynchronizations.

4 Design Considerations

At the heart of our proposed implementation of the adopted pet protocol is a pseudo-random bit generator implemented (as a finite-sate machine – FSM) on a passive RFID tag. The exact details of the generator are not specified in the protocol. In fact, the protocol assumes that these details are secret.

Pseudo-random bit generators are heavily used in cryptography for *stream ciphers.* When used in this way, the two parties to the communication must utilize exactly the same generating algorithm and agree on a shared secret—the encryption key. Usually this means that the details of the algorithm such as the characteristic polynomials and the filtering function or combining function, are public knowledge and the encryption key takes the form of the initial state of the generator.

However, that is not how we will be using the generator. In our context, there is no shared secret, and in fact, the state of the generator is not all that important. Instead, we treat the specific parameters of the generator as secret. Each tag will have its own set of parameters. In our realization of the AP protocol, *the RFID tag's key is the characteristic polynomial.* In order to gain the trust of the tag, the reader must prove that it knows the polynomial, either by sending it to the tag, or by predicting additional bits of the sequence and sending those to the tag. This latter "proof" is desirable if the tag is not aware of the characteristic polynomial associated with its internal FSM.

We shall discuss several implementation options in the following subsections. For now, we review some of the simple facts behind pseudo-random bit generators. Let $z_0, z_1, z_2, \ldots$ be a sequence of bits generated by the tag. As it is generated by a finite state machine, the sequence is eventually periodic. Consequently, it is possible to recreate the sequence via a linear recurrence relation such as

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

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 polynomial

$$f(X) = c_0 + c_1 X + c_2 X^2 + \cdots + c_{L-1} X^{L-1} + X^L \tag{2}$$

is called the *characteristic polynomial* of the sequence.

The following well-known fact is relevant to this discussion.

Lemma 1 *The characteristic polynomial of a sequence of linear complexity L can be computed from $2L$ consecutive bits of the sequence.*

Further discussion of this issue is postponed until Section 5.1 because it concerns not only the legal owner of the tag, but also an eavesdropper who may wish to inappropriately gain the trust of the tag. In the following subsections we discuss some potential designs.

4.1 The Bare Linear Feedback Shift Register

So far, in the family of finite-state machines we have discussed only the LFSR. The LFSR is a good candidate for our AP protocol, but may not be able to satisfy all foreseeable demands. Take the following example.

Example 2 *Consider a system with the same privacy characteristics as that in Example 2 (10 hours minimum uninterrupted time spent with a reader before pairing, and 5 hours minimum uninterrupted time before any information is leaked). However, assume that the tag is required to respond to interrogations at least once every second (instead of once every minute as in Example 1). The tag's internal LFSR should have length $L = 18000$.*

Such an LFSR is too long for practical purposes. As discussed in Section 6 below, the extremely stringent cost constraints characteristic of RFID technology do not allow the use of LFSRs of total length more than 150 for security purposes. This value is only half the length we used in our Example 1, and less than one hundredth of the length required by Example 2.

However, implementing the generator as a simple LFSR of length (and, for a primitive characteristic polynomial, linear complexity) $L \leq 150$ is almost surely too low for reasonable security.

Take-Away Point 6 *To conclude, our protocol calls for a pseudo-random bit generator with linear complexity L of moderate size (in the order of 10^4), involving registers with cummulated length less than 150. The bit-generation rate and linear complexity should be such that the legitimate owner can collect $2L$ consecutive bits under normal circumstances, while an eavesdropper is unlikely to collect even an L-bit subsequence in each attempt.*

Several methods are available for increasing the linear complexity of a LFSR-based pseudo-random bit generator. In what follows, we discuss three of the most popular ones.

4.2 The Nonlinear Combination Generator

A *nonlinear combination generator* is composed of N LFSR's operating in sync. At each step, each of the shift registers generates a new output bit simultaneously – let these outputs be $x_n^1, x_n^2, \ldots x_n^N$. The output of the generator is obtained by applying a nonlinear combining function f to the outputs of the component LFSR's: $z_n = f(x_n^1, x_n^2, \ldots x_n^N)$

The period of the resulting sequence $\{z_n\}$ will be the least common multiple of the periods of the component generators. If all of the component LFSR's have primitive characteristic polynomials, and if the degrees of those polynomials are pairwise distinct and approximately equal to L_0, then the linear complexity of the output sequence will be in the order of L_0^r, where r is the nonlinear order [11, 6.48] of the function f [11, 6.49].

However, while the bare LFSR was only subject to linear-complexity attacks, the more sophisticated nonlinear combination generators are also the subject of correlation attacks. In fact, their vulnerability against correlation attacks is a direct function of the correlation immunity of f [11, 6.52]. It is also known [11, 6.53] that if the nonlinear function f is m-th order correlation immune, and balanced, then its nonlinear order is bounded as $r \leq N - m - 1$. For a guide to constructing correlation-immune functions of a given order, the reader is refered to [12].

In the remainder of this section, let us consider that such a nonlinear m-th order correlation immune function f, of nonlinear order $N-m-1$ is readily available. The linear complexity L of the nonlinear combination generator employing f is $O(L_0^{N-(m+1)})$, the total register length is NL_0, and the complexity of a correlation attack is $O(2^{2L_0(m+1)})$ [11]. We consider that resistance to correlation attacks of order less than 2^{128} is adequate for our application.

Example 3 *Imposing our design constraints, we get $NL_0 < 150$ and $2L_0(m+1) > 128$, which result in $N < 150/L_0$, and $(m+1) > 64/L_0$. The linear complexity is thus $L < L_0^{86/L_0}$. To accomodate the requirements from Example*

2, we can set $L_0 = 30$, yielding a linear complexity $L \simeq 17000$, and requiring a number of $N = 5$ registers (of length L_0 each), and a first-order correlation immune function f of nonlinear order $m = 3$.

It is worth noting that the function $L(x) = x^{a/x}$ with $x > 0$ has first-order derivative

$$\frac{dL(x)}{dx} = x^{a/x} \frac{a}{x^2} (1 - \ln(x)), \tag{3}$$

which implies the existence of a single (global) maximum at $x = e$ (Euler's number). Hence, for discrete $x = L_0$, the maximum is always at $L_0 = 3$, and roughly equal to $e^{a/e}$. Since in our scenario $a = (N-m-1)L_0$, it is possible that for certain design constraints (small total register length or high complexity of the correlation attack), the desired linear complexity is not achievable. Also, when the desired linear complexity is achievable, there will generally be two possible choices for L_0. For implementation purposes, the value larger than 3 should be desirable.

4.3 The Nonlinear Filter Generator

A *nonlinear filter generator* consists of an LFSR of length L_0 and a nonlinear filtering function F of nonlinear order r. Starting from an initial seed (or state) $x_0, x_1, \ldots, x_{L_0-1}$, the LFSR determines additional bits according to the recurrence relation

$$x_{n+k} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{L_0-1} x_{n+L_0-1}.$$

The output of the generator is the sequence $z_0, z_1, \ldots$ determined by the rule $z_n = F(x_n, x_{n+1}, \ldots, x_{n+L_0-1})$.

The period of the output sequence is the same as that of the LFSR, which can be set at $2^{L_0} - 1$ by choosing the characteristic polynomial of the LFSR to be primitive. The linear complexity of the output sequence can be bounded as [13]:

$$\binom{L_0}{2} \leq L \leq \sum_{j=1}^{r} \binom{L_0}{j}, \tag{4}$$

where r denotes the degree of the polynomial F (i.e. F's nonlinear order).

The nonlinear filter generator is subject to multiple types of attacks, in addition to the linear-complexity attack. In what follows, we discuss these attacks and show that none of them is a serious concern for our application. The inversion attacks of [14, 15] rely on the public knowledge of the component LFSR structure and of the nonlinear filtering function (which in our AP protocol are the secrets of the tag), and can be foiled anyway by selecting a proper generator design. Another class of attacks, which also require complete knowledge of both the characteristic polynomial of the LFSR and the filtering function, are the *algebraic attacks* of [16].

Several types of correlation attacks have been investigated (see [12] for a literature review), but they all assume publicly-known LFSR connection polynomials. It is interesting to note that some attacks, which assume that the nonlinear function is also secret, will first attempt to find an equivalent nonlinear combination generator [12, 17].

For our application, the fact that the nonlinear filter generator uses a single LFSR has a certain advantage over the other multiple-LFSR pseudorandom bit generators like the nonlinear combination generator discussed in Section 4.2. To see this, note that the divide-and-conquer correlation attacks devised for the nonlinear combination generator will brute-force the initial state of one LFSR at a time. Hence, they can be easily adapted to deal with unknown LFSR characteristic polynomials by brute-forcing these polynomials, along with the initial states, one LFSR at a time. However, the single LFSR in the nonlinear filter generator can be designed with a register length of roughly $L_0 = 150$ (if we neglect the resources required for the implementation of the nonlinear function). Note that this value of L_0 guarantees a linear complexity at least in the order of 10^4, which is what was required in Example 2. However, adapting the existing attacks to an unknown characteristic polynomial is no longer practical – brute-forcing the single, longer LFSR would increase the complexity of the attack by $O(2^{150})$.

4.4 The Shrinking Generator

A *shrinking generator* [18] consists of two registers of lengths L_A (for the *A-sequence*, or the output-generating sequence) and L_S (for the *S-sequence*, or the controlling sequence), controlled by the same clock. The generator only produces an output bit at time n if the n-th bit of the control sequence is 1. In this case, the output of the generator coincides with the n-th bit of the output-generating sequence. However, if the n-th bit of the control sequence is 0, the generator does not output anything. The linear complexity L of a shrinking generator defined as above was bounded in [18] as $L_A 2^{L_S-2} \leq L \leq L_A 2^{L_S-1}$.

A variant of the shrinking generator, the "self-shrinking generator", consists of a single LFSR. The odd bits of the LFSR's output function as the *A-sequence*, while the even bits work as the *S-sequence*. The linear complexity of the self-shrinking generator with a register of length L_{SS} was shown in [19] to be bounded as $2^{\lfloor L_{SS}/2 \rfloor -1} \leq L \leq 2^{L_{SS}-1} - (L_{SS} - 2)$.

Looking back to Example 2, a linear complexity of $L = 18000$ could be achieved by a self-shrinking generator of length only $L_{SS} = 32$, or by a shrinking generator with two registers of length $L_S = L_A = 13$.

However, it turns out that the easily-achievable high linear complexity of the shrinking generator and its variants comes at a cost. It was shown in [18] that the shrinking generator (with secret connection polynomials) is subject to a correlation attack that requires only $2L_A L_S$ contiguous bits and time at most $O(2^{2L_S} L_S L_A^2)$ [18]. If we recall that $L \geq L_A 2^{L_S-2}$, we see that in order to achieve a reasonably high computational complexity (which would render the protocol secure from an attacker)—a complexity in the order of 2^{128} would be acceptable—we need to increase the linear complexity to values much larger than our AP protocol can deal with. Recall that our protocol relies on the legitimate user to mount a successful linear-complexity attack.

Example 4 *Take $L_A = L_S = L_0$. Our design constraints can be written as $2L_0 < 150$ and $2^{2L_0}L_0^3 > 2^{128}$. The latter one implies $L_0 > 56$, which requires a total register length of $2L_0 > 112$ (this would satisfy the first constraint), but yields a linear complexity $L > L_0 2^{L_0-2} \simeq 10^{18}$. For a legitimate reader to mount a successful linear-complexity attack within 10 hours of uninterrupted time, it would need to querry the tag more than 10^{13} times per second. Note that the highest frequency band allowed for RFID applications is around $10GHz$ (UHF-RFID)[1], and hence RFID applications cannot accommodate a "baseband" signal with a bandwidth of $10GHz$.*

Take-Away Point 7 *To conclude this section, we have seen that our AP protocol can be implemented at very low cost, by using a nonlinear filter generator, or a nonlinear combination generator. While a bare LFSR could be adequate for certain low-security-risk applications (like RFID tags attached to groceries) where a small linear complexity can be tolerated, the shrinking generator increases the linear complexity more than most applications can handle (it could take years of uninterrupted UHF communication for a reader to mount a successful linear-complexity attack against a correlation-immune shrinking generator).*

5 Security Analysis

Based on the discussion in Section 4, we can state that the AP protocol implemented with properly-designed nonlinear combination generators, or nonlinear filter generators, is immune to correlation attacks. Hence, it seems that the best technique for an attacker to gain control of the tag is to follow the same process as a legitimate reader: to query the tag and collect a number of bits large enough to enable a linear-complexity attack. In this section we consider this attack from the malicious user's perspective, and show that it is not feasible under normal conditions. In addition, we discuss whether or not the attacker could extract more information from the knowledge of the tag owner's daily routine.

5.1 Linear-Complexity-Based Attacks

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:

$$\begin{aligned} c_0z_0 + c_1z_1 + \cdots + c_{L-1}z_{L-1} &= z_L \\ c_0z_1 + c_1z_2 + \cdots + c_{L-1}z_L &= z_{L+1} \\ c_0z_2 + c_1z_3 + \cdots + c_{L-1}z_{L+1} &= z_{L+2} \\ &\vdots \\ c_0z_{L-1} + c_1z_L + \cdots + c_{L-1}z_{2L-2} &= z_{2L-1}, \end{aligned} \tag{5}$$

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 $O(L^3)$. This observation constitutes a proof of Lemma 1.

Note that in the previous paragraph, the linear complexity of the generator 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 [9] 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.

However, to formulate and solve the system in (5), we implicitly made the following two assumptions: (A) $2L$ bits of the sequence are known, and (B) 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.

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 (5), 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 [20], 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 [21] 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 $O(2^{\min(L,k)})$.

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 3 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.

Take-Away Point 8 *The considerations above show that, for a legitimate reader that should have access to a large number of contiguous bits, with small interruptions, finding the secret key should be feasible, by employing a simple exhaustive-search over the missing bits, and solving the system in* (5). *However, if the interruptions are large and the sequences of contiguous bits small (the case of the "man on the bus"), the secret key is very likely to remain secret.*

5.2 Predictable Environments and Tag Tracking

We have already specified that our protocol will function in a *ubiquitous RFID environment*. This assumption is not at all restrictive: it is meant to create a challenging environment for the legitimate user, rather than hinder the attacker. Whether the assumption holds true or not, the tag is being interrogated numerous times throughout the day, either by various non-authenticated readers, or by the legitimate reader alone. Therefore, it is safe to assert that the number of bits produced by the tag's internal LFSR between two attacker sessions is unknown and unpredictable.

However, let us suppose that the number of bits produced by the tag's LFSR between any two attack sessions is roughly the same (a daily routine, along with a very punctual attacker might enable this scenario). A natural question is whether the attacker can synchronize his encounters with the tag owner in such a manner that would grant him access to consecutive short spans of the tag's output. For example, the attacker might be able to obtain S bits of the tag's output at each encounter. If the attacker synchronizes the encounters such that the tag has generated exactly $T+S$ bits between two encounters, where T is the period of the pseudo-random sequence generated by the tag's secret FSM, the bits obtained in two consecutive encounters would be consecutive. Even if the actual number of bits generated between two encounters a random variable R with mean $T+S$ and very small variance, the attacker can assume that $R{=}T+S$, and attempt to correct for the error in this assumption by using an exhaustive-search method of the type described in the previous section. We shall call this strategy a *"predictable-environment attack"*.

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 sequences of S bits that an attacker might be able to gather in his lifetime belong to the same LFSR period.

Take-Away Point 9 *Consequently, any "predictable-environment attack" is reduced to the intractable large-desynchronization problem of Section 5.1.*

Another problem which may arise in a predictable environment is *tag tracking*. Consider a scenario where the attacker acts as the man on the bus for a while (possibly multiple encounters), and then wants to use the information gathered about the targeted tag to verify the presence of its owner in a certain spot, at a certain time. For example, gaining access to the RFID readers in a bar, the man on the bus wants to find out whether the victim was present there last night. The problem is quite complex, and is beyond the scope of this paper. However, for completeness, in the remainder of this subsection, we briefly discuss a worst-case type of scenario.

Assume that the attacker has gained access to S_1 consecutive bits (highly unlikely) of the targeted tag's output on the bus, and collects another S_2 consecutive bits from the bar's RFID reader. Let us also assume that the attacker knows that, if the S_2 bits were produced by the targeted tag, then there would be exactly r unknown bits between the bus encounter and the moment the bar RFID reader began gathering bits from the tag (that is, the tag was interrogated r times by other, unknown readers). It is clear that if $S_1 + S_2 < 2L$, the attacker can make no inference about whether the S_2 bits were produced by the targeted tag. This is due to the fact that the attacker's best strategy is to compute a system of $S_1 + S_2 + r - L$ (quadratic) equations with $L + r > S_1 + S_2 + r - L$ unknowns, as explained in Section 5.1 above. Such a system would likely admit an infinity of solutions. On the other hand, if $S_1 + S_2 \geq 2L$, the attacker can compute a system of more than $L + r$ equations, with $L + r$ unknowns. If the system has no solution, the attacker can infer that the S_2 bits obtained from the bar were not produced by the targeted tag. However, a more involved probabilistic analysis is required if the system does admit a solution. For example, if $S_1 = 1$, the fact that the system has a solution does not provide much certainty about the presence of the targeted tag in the bar. Neither does the case when S_2 is small. On the opposite end, if $S_1 \geq 2L$ and $S_2 \geq 2L$, then the attacker can verify the presence of the targeted tag in the bar with absolute certainty. Such an analysis is beyond the scope of this paper.

Nevertheless, for practical situations, the worst case scenario would still have the attacker check whether a system of $S_1 + S_2 + r - L$ quadratic equations with $L + r$ unknowns has a solution. This problem is as hard as finding the solution, and would normally prevent the attacker from attaining his goal, as discussed in Section 5.1 above.

6 RFID Hardware Constraints

In Section 4.1 we have already mentioned that we can only afford to implement the AP protocol using LFSRs of cummulated length less than 150. To see why, we outline the design limitations imposed by the passive RFID architecture. It seems to be generally accepted that a reasonable resource expense for security should not exceed 2000 transistors for a passive RFID tag. Since most pseudo-random bit generators are based on linear feedback shift registers, let us consider the implementation of an LFSR on such a tag. Each cell in an LFSR is a latch with 8 transistors (2 NAND or 2 NOR gates), and each binary addition is an XOR gate with 5 transistors. This brings the total number of transistors used by an LFSR of length L_0 to roughly $13L_0$. Based on our ceiling of 2000 transistors, this limits the number of cells to 150.

Moreover, if the AP protocol implementation dictates the use of a nonlinear function for combining the outputs of several LFSRs (or filter the state of a single LFSR), the design should consider registers of total length even less than 150, to allow for the implementation of the function.

7 Concluding Remarks

We have introduced a novel pairing protocol, ideally suited for low-cost, passive RFID devices. Our protocol is entirely automatic, and based on the two pairing devices spending large periods of uninterrupted time in the proximity of each other. Successful pairing is based on the legitimate reader mounting a liner-complexity attack on the tag's internal pseudo-random number generator (PRNG), where the tag's secret key is the internal structure of the PRNG (or more precisely, the characteristic polynomial of the equivalent LFSR). The algorithm design prevents an unauthorized user from learning the tag's secret, and hence from pairing with the tag. This level of security is ensured by the fact that any unauthorized user should not be able to spend an uninterrupted period of time larger than several hours a day in the company of the tag.

The examples provided throughout the paper show that practical implementation of the tag's PRNG should consider either nonlinear combination generators or nonlinear filter generators. We have shown that, while simple LFSRs may be considered for certain applications, more powerful linear-complexity-enhancing designs, like the shrinking generators, could render the linear-complexity attacks (on which the legitimate user relies) unfeasible.

The Adopted Pet protocol is a first step towards a paradigm where authentication and security is based on the legitimate parties mounting successful attacks on each-other's cryptographic protocols, and where the work of anonymous attackers and hackers can serve as the basis for faster authentication and legitimate decryption.

8 Acknowledgement

This work was partially supported by NSF under grants No. CNS-0644238, CNS-0831470, and DUE-0911708. We appreciate anonymous reviewers for their valuable suggestions and comments.

References

1. Dobkin, D.M.: The RF in RFID: passive UHF RFID in Practice. Elsevier Inc. (2008)
2. Stajano, F., Anderson, R.: The resurrecting duckling: security issues for ad-hoc wireless networks. AT& T Software Synposium (1999)
3. Mayrhofer, R., Gellersen, H.: Shake well before use: authentication based on accelerometer data. Pervasive Computing **4480** (2007) 144–161
4. Oshiba, T., Nebayashi, H.: Device pairing based on adaptive channel fluctuation control for large-scale organizations. ISCC 2009. IEEE Symposium on Computers and Communications (2009) 901–906
5. Soriente, C., Tsudik, G., Uzun, E.: BEDA: button-enabled device association. International Workshop on Security for Spontaneous Interaction (IWSSI07) (2007)
6. Luo, S., Xia, H., Gao, Y., Jin, J., Athauda, R.: Smart fridges with multimedia capability for better nutrition and health. International Symposium on Ubiquitous Multimedia Computing, 2008. UMC '08. (2008) 39–44
7. Gu, H., Wang, D.: A content-aware fridge based on RFID in smart home for home-healthcare. 11th International Conference on Advanced Communication Technology, 2009. ICACT 2009. **2** (2009) 987–990
8. Hopper, N.J., Blum, M.: Secure human identification protocols. Advances in Cryptology ASIACRYPT 2001 **2248/2001** (2001) 52–66
9. Massey, J.L.: Shift-register synthesis and BCH decoding. IEEE Trans. Information Theory **15** (1969) 122–127
10. Juels, A.: Minimalist cryptography for RFID tags. Security of Communication Networks (SCN) (2004)
11. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press (1996)
12. Cusick, T.V., Stanica, P.: Cryptographic Boolean Functions and Applications. Elsevier Inc. (2009)
13. Lhlein, B.: Attacks based on conditional correlations against the nonlinear filter generator. In: In Cryptology ePrint Archive, Report 2003/020. (2003)
14. Golic, J.D., Clark, A., Dawson, E.: Generalized inversion attack on nonlinear filter generators. IEEE Trans. Computers **49** (2000) 1100–1109
15. Golic, D.: On the security of nonlinear filter generators. Proc. Fast software encryption – Cambridge'96 (1996) 173–188
16. Courtois, N.T., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: Advances in cryptology—EUROCRYPT 2003. Volume 2656 of Lecture Notes in Comput. Sci. Springer, Berlin (2003) 345–359
17. Siegenthaler, T.: Cryptanalysts representation of nonlinearly filtered ml-sequences. In: Proc. of a workshop on the theory and application of cryptographic techniques on Advances in cryptology—EUROCRYPT '85, New York, NY, USA, Springer-Verlag New York, Inc. (1986) 103–110

18. Coppersmith, D., Krawczyk, H., Mansour, Y.: The shrinking generator. Advances in cryptology (CRYPTO'93) **773** (1993) 22–39
19. Blackburn, S.R.: The complexity of the self-shrinking generator. IEEE Transactions on Information Theory **45** (1999) 2073–2077
20. Fraenkel, A.S., Yesha, Y.: Complexity of problems in games, graphs and algebraic equations. Discrete Applied Mathematics **1** (1979) 15 – 30
21. Courtois, N.T., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: Advances in cryptology—ASIACRYPT 2002. Volume 2501 of Lecture Notes in Comput. Sci. Springer, Berlin (2002) 267–287 Updated version: `http://eprint.iacr.org/2002/044`.