Technical mechanisms for RFID privacy

cti.gr

Abstract

When clothing manufacturer Benetton announced in March 2003 that it was considering the use of Radio Frequency Identification (RFID) chips in its garments in order to streamline its supply chain, an unexpected storm of protest followed in the media that ultimately forced the company to withdraw its plans only a few weeks later. Ever since, RFID technology has become one of the most debated ubiquitous computing technologies, and public fears of its alleged capability for comprehensive surveillance have prompted a flurry of research trying to alleviate such concerns. The following article aims at introducing and briefly evaluating the range of proposed RFID privacy solutions so far.

Introduction[1]

RFID tags represent a significant privacy problem – at least in principle – due to their enhanced means for identification. While proponents of this technology often like to compare RFID to the ubiquitous, yet by no means threatening bar codes, RFID does differ from them in two important respects:

1. Level of Detail: While special two-dimensional versions might carry up to 1000 bytes at the expense of larger print areas and lower reliability during scanning, the majority of today's barcodes feature only about a dozen digits. RFID tags in contrast store usually hundreds bits, and are already designed to carry not only a class-identification (e.g., a manufacturer-id and product-id) but rather item-level-identification (i.e., a serial number). Some types of RFID tags can even be rewritten.

2. Unobtrusiveness: Reading a barcode requires a line-of-sight between the reader and the tag. This means not only that the scanning process itself can hardly go unnoticed, but also that the tags must be easily visible. RFID-Labels in contrast are read from (or written to) through an electromagnetic field, which can easily penetrate plastic, fabrics, or paper. Thus, both the fact that a tag is present, as well as the act of reading out such an RFID-tag can be concealed.

Work on technical privacy-protection tools for RFID-tags has therefore focused on reducing the amount of detail reported by such tags, e.g., by replacing the stored serial number with a generic manufacturer code or even a completely arbitrary number, and on preventing any unnoticed read-outs of such tags. Due to the envisioned widespread usage of such tags, the former method might only be a partial solution: Even if the level of detail provided by such tags is significantly reduced, the specific combination of tags carried by an individual, socalled "constellations" [1], might still allow for the identification of a person. Existing technical solutions in the field of RFID privacy can be divided into anonymizing and pseudonymizing methods. Both can either be achieved by deleting or altering the data on the tag itself, or by controlling read access to it. Especially the latter is critical, since RFID readers must also provide the energy to power the battery-less tags, resulting in reader-to-tag communication that stretches much further than the corresponding return channel from the tag back to the reader.

The RFID kill-command

Long before the Benetton incident triggered a public controversy over the use of RFID tags in consumer articles, the 2002 Auto-ID specification[2] contained the requirement of a "kill"-command [2]. The basic idea is simple: Before selling a tagged item to the consumer, the embedded tag is permanently deactivated at checkout. This renders the tag inaccessible to subsequent reader commands and thus prevents any tracking or profiling beyond the point of sale.

The current Auto-ID/EPC global specification[3] requires for all conformal tags an 8-bit-password to be set on the tag during or right after production in order to prevent unauthorized deactivation of the tags through this kill-command, e.g., while still on the shelves. After receiving the correct password, the specification requires the tag to stop responding to all subsequent reader commands in any way [3]. How this functionality is actually implemented on the tag is left up to the manufacturer, though due to cost efficiency, most solutions are currently software-based, which would allow – at least in principle – a later reactivation of the tag through direct contact (as the over-the-air interface is deactivated).

Apart from this potentially incomplete tag destruction, two additional aspects significantly affect the efficiency of this method from a privacy point of view. For one, deactivating the tags at checkout would still allow for detailed tracking of consumers inside stores, as well as associating consumer data and shopping information right at the point of sale (e.g., through the use of a credit or consumer card when paying). Additionally, the process of deactivation itself is for the consumer difficult to verify, as no visible cues would be present. The fact that all known deactivation methods are software-based, even though a permanent electro-magnetic deactivation similar to today's anti-theft labels would in principle be equally possible, is seen by critics as further proof that a later reactivation is left as a possibility – a suspicion that seems to have been already vindicated by some fielded prototypes: during a visit to Metro's Future-Store by RFID-activist Catherine Albrecht, a detailed inspection of the supposedly killed tag revealed that only Metro's own product number had been deleted, while the tag's hardware serial number was still left intact due to "technical reasons" [4].

Others point out that equipping all existing point of sales with "kill stations" is widely unrealistic [5], since small businesses such as kiosks would never be able to afford the corresponding equipment, even though they would inevitably sell tagged merchandise (e.g., soda cans or razor blades). Today's prototypes for tag deactivation are also not yet capable of handling multiple tags at once: not least due to the password-protection mechanism, customers must laboriously silence each individual tag manually – a nuisance that might prompt many customers to abstain from bothering with the deactivation.

Permanently deactivating tags of course also prevents any secondary use of such identifiers, e.g., as part of the often-cited intelligent fridge or other smart household appliances; for providing follow-up services such as automatically recommending matching accessories for tagged clothing; and to improve product life-cycle services such as repairs, returns, and recycling. A comprehensive use of RFID even after the point of sale would benefit not only manufacturers and retailers in the form of an increased consumption through countless smart fridges, but also consumers, who might appreciate being told of expiring produce, or to be able to simply return a defective product without having to worry about keeping the receipt (since the product's RFID tag stored all relevant data for the return).

Hash locks and metaIDs

As an alternative to the "all or nothing" approach of the kill command, a number of proposals favor protecting the RFID-tag payload (i.e., the tag ID or, alternatively, the stored electronic product code) from unauthorized reading. As soon as a product changes into the hands of the consumer, a key-based access protocol would allow him or her to control who would be allowed to subsequently read out the stored tag information.

The basic principle was already proposed in 2002 by Sarma et al. [6] and is based on mathematical one-way-functions, so called "one-way hashes." In order to "lock" an RFID-tag, an RFID-reader device would choose an arbitrary key k, compute a hash value $h = H(k)$ from it using a reasonably secure one-way hash function, and store this hash value (called the "MetaID") in a specially reserved area on the RFID-tag. In order to facilitate unlocking the tag at a later time, the owner (or better: his or her tag-reader) would also file the random key k under its MetaID h in a database under the owner's control. Once a tag has a stored MetaID, it replies to all read requests with only this MetaID, never with its "true" ID or any other data payload it might carry (e.g., its Electronic Product Code (EPC) – a standardized identifier that not only carries a manufacturer and product ID, but also a product's serial number). If the tag owner later wants to access the original data again, he or she simply queries the tag for its MetaID h (which is the only information accessible from the tag, anyway) and looks up the corresponding key k that was originally chosen by the reader to lock the tag, using the database of key-MetaID pairs. Once this key k is sent to the tag in question, the tag will itself perform the computation of H(k) and verify if it matches its stored MetaID. Should this be the case, it deletes the stored MetaID and is thus effectively unlocked again.

An access-control scheme using one-way hashes as keys has several advantages. Even though it does not offer absolute security in the mathematical sense, computing the original unlock value k from the stored hash value h requires such a substantial effort that for all practical purposes, being able to read out the MetaID h will not allow an unauthorized reader to deduce the original value k for unlocking the tag. Also, providing RFID-tags with the ability to compute a hash-value (for verifying that the reader-sent unlock value k does indeed form the basis for the stored MetaID h = H(k)) is relatively cheap to implement [1], and would thus also be an option for ultra-cheap RFID-tags – an important advantage over more complex (and therefore potentially more secure) solutions that use symmetrical or asymmetrical cryptography, which are only an option for relatively expensive goods that can "afford" an expensive tag.

An improvement of such static MetaIDs is the use of so-called "randomized hash-locks" [7]. Their goal is to prevent the creation of detailed tracking records by repeatedly accessing a fixed MetaID using several different reader devices. For this, tags do not reply with a fixed MetaID anymore, but instead generate their MetaID anew upon each read request from a tag-reader. An integrated random number generator on the tag generates a random value ri, which is appended to the "real" ID of the tag and thus forms the basis for a temporary MetaID hi = H(ID||ri). A reader receives both the temporary hi as well as the used random number ri. In order to deduce the real ID of the tag, the reader needs a list of all possible IDs – a requirement that seems feasible for individuals with a small number of tagged items (as opposed to large supermarkets with hundreds of thousands of tagged items in store). Using this list of known items, the reader device then simply computes hj = H(IDj ||ri) for all its known IDs, until it finds an hj that matches the hi it read from the tag. With this, it implicitly knows the ID of the tag and does not even have to explicitly unlock the tag (which would work analogous to the fixed MetaID scheme). Only if an item would be returned or transferred to a different individual, the reader would send the found “true” tag-ID IDj and thus unlock the tag again.

Access control

A different approach to authenticating legitimate reader devices is put forward by Fishkin and Roy [8]: Based on the principle distance implies distrust, Fishkin and Roy propose tags that return more or less information based on the distance to the reader devices that poses the query. As an example, they list five possible levels of disclosure: At level zero, the tag only announces its presence. At level one, it replies with generic attributes (e.g., a shirt would reply with its color and fabric). Only at the highest level of four, personally identifiable information such as the location and time of purchase would be released.

While the basic principle of their approach is rather simple, the practical implementation is not. At the outset, the signal strength of a reader device at a tag depends heavily on the tag’s orientation – as soon as it changes from its “optimal” position, the reader will appear much further away than it really is. While this might be tolerable from a privacy point of view (after all, more distance implies less data transfer), it would make reliable application design almost impossible[4]. Additional, both metallic substances and water[5] significantly influence the energy field of an antenna, which makes reliable measurements outside laboratory settings difficult. While the authors hope to increase reliability by combining the different approaches, and by putting more complex antennas on the tags, the difficult “user interface” of such a solution, as well as its increased cost, will most likely appeal neither to customers nor to service providers. That is because even with a reliable distance measurement, consumers would be unable to judge the actual information exchanged in everyday operations, where, in theory, leaning too close to a (potentially unknown) reader could accidentally disclose detailed information. This also prompts the question whether the hierarchical organization of tag-data is always useful or even possible.

Eavesdrop-resistant anti-collision protocols

Due to the power asymmetry between reader and tag, information sent from reader devices would be subject to eavesdropping, even if using one of the above authorization methods, where only “friendly” reader-devices would get access to the information stored on the tags. This is because of the energy field of the reader, which not only transmits the information from the reader to the tags, but is also used to power them, and thus typically has a much larger range than the signal that is reflected back from the tag. This allows third parties to “listen in” on the signal sent from the reader, even from a considerable distance.

This is especially critical if the tag’s ID is among the information sent from the reader to the tag. While this might sound unlikely at first (after all, it is the reader that is interested in the tag ID, not the other way around), it is quite common practice in binary-tree-based anti-collision protocols [9]. As tags typically have no way of detecting the presence of other tags, their replies to a reader’s signal might conflict with the signals from other tags in the vicinity, thus creating a “collision,” an interference that prevents the reader from decoding the IDs of all of the involved tags.

A popular variant of such a protocol uses ID prefixes sent from the reader to determine which tags (i.e., only those with a common prefix) should reply. As long as the reader detects a collision (i.e., if two or more tags with the same prefix as indicated by the reader are within range), the reader increases the length of the prefix (e.g., by adding a “1” to it) until a single tag ID can be “singularized.” It then replaces the bit it added last with its inverse and continues – should more collisions occur – to increase the length of the prefix. This explicit partitioning allows the individual selection of an arbitrary number of tags. However, the above asymmetric transmission power would allow a third party to log the sent-out prefixes, potentially learning the individual tag IDs should a collision occur at the very last bit position.

Weis et al. [7] propose that instead of sending a whole prefix, readers would only send the command “transmit next bit” to the tags. As long as their corresponding bit positions are identical, no collision would occur[6] and the reader would be able to note the common bit prefix incrementally. Once two tags would differ at position i, the reader would just as before use a “select” command to pick a subtree, but instead of sending the complete prefix to the tags (i.e., sending bits 1 through i, with either “1” or “0” at position i), it would simply XOR Biti−1 with its chosen Biti and send the resulting value. Tags in turn would XOR the received bit with their own Biti−1 (which must be identical to the reader’s Biti−1) and compare the resulting value to their corresponding Biti. In case of a match, a tag would be selected and reply with its Biti+1. An attacker who could only listen to the forward channel (i.e., who could “hear” the commands of the reader, but not the replies from the tags) would not be able to observe the bits of collision-free prefixes (since the reader only sends a “Send next Bit”-command and the replies from the tags are too weak to be detected over long distance). Similarly, such an attacker would be unable to deduce any bit-values in case of collisions, as the XOR with an unknown value (Biti−1) also hides the reader-selected subtree-bit at position i[7]. However, in order to “remember” the current bit position, tags would need to carry (expensive) dynamic memory.

An alternative anti-collision method can potentially work without sending out any information on the forward channel: In protocols based on the Aloha-Model, tags reply individually with a random delay to the reader signal [10]. Depending on the (reader-set) time allocated for tag-replies, tag transmissions distribute themselves randomly and can ideally be read collision-free. However, in order to increase the performance of such protocols, some variants explicitly “silence” tags that have been correctly identified, in order to lessen the number of tags that need to be read if only a few collisions occur. Unless special care is taken, such a selection mechanism would of course allow a distant attacker to log the IDs of such silenced tags.

The current EPCglobal tag specification [3] contains a requirement for a random-number generator on the tag, both for reasons of efficiency and security. Instead of its “true” ID (typically the EPC), the specification requires tags to reply with a random number that is generated for each read cycle anew. In order to “silence” a tag under this protocol version, the reader uses this random number. Once all tags have been identified using their momentarily chosen temporary IDs, readers can then use these numbers to request the “real” ID from each tag. This not only prevents attackers from “listening in,” but also increases the speed of the anti-collision protocol as the temporary ID uses fewer bits (12) than the globally unique EPC (96) and thus provides for shorter transmission times[8].

The blocker-tag

Probably the simplest proposed access control method for RFID-tags is based on the above described binary-tree-based singularization protocol and follows a denial of service approach [11]. Juels and Pappu propose that consumers carry a so-called blocker-tag with them, which replies to any read request with a self-induced collision (using two antennas that reply with two conflicting IDs). Using the above mentioned binary-tree- based anti-collision protocols, readers would thus begin the task of singulating individual tags from the apparently large population of tags. However, for any prefix sent from the reader device, the blocker-tag would create a collision, therefore forcing the reader to traverse the entire tree of all possible ID combinations – when using a 96-bit EPC, it would have the size of several billions of tags. Even if a reader would be able to read several thousand tags per second, the presence of such a blocker-tag would effectively stall any read attempt indefinitely (or until the reader device would give up)[9].

The biggest advantage of the blocker-tag approach is certainly the minimal infrastructure that is needed: existing tags (at least those with rewritable memory) could be used unchanged, and reader devices would only need minimal software updates to cope with privacy zone announcements. On the other hand stands the rather poor reliability of such a method: by implementing blocker-tags cheaply as a passive RFID-tag, a slight misalignment could easily cut power to the blocker-tag and thus expose the formerly hidden tag population. Using cheaper, non-writable tags would keep costs further down, yet would greatly increase the interferences between blocker-tags and legitimate read operations: A neighbor helping with the shopping bags prevents my smart fridge to detect half of my groceries, and my smart laundry machine is unable to detect the proper program due to the blocker-tag I left in the pocket of my jeans. Equally possible seem advancements in reader technology that would allow readers to differentiate between “real” collisions and those that are simulated with a blocker-tag.

Summary

RFID is probably one of the most prominent ubiquitous computing technologies today, owing to its widespread use (or planned use) in industry and its direct effect on consumers. The traditional, security-only based privacy solutions presented in this section often fail to be practically viable: Fishkin and Roy’s distance-based authentication principle [8] seems appealing due to its intuitive simplicity (“distance implies distrust”), though it is most likely infeasible to realize technically, let alone reliably controllable for the consumer. Blocker-tags [11] are equally unreliable, as a slight misalignment of the blocker tag can quickly reveal the entire protected tag population.

More reliable and robust are the proposed hash-lock and MetaID mechanisms [6], which make involuntary data disclosures unlikely as the “real” ID of an item is never revealed. However, MetaID solutions require not only a more complicated infrastructure setup, but are also not able to prevent tracking attacks using “constellations” of tags. Variable MetaIDs [7] remedy this by providing a different number on every read, yet greatly increase overall system complexity, as all ID changes need to be tracked in a database. Also, users will need to engage in detailed tag management in order to properly register or unlock tags for the various applications they are allowed to work in (e.g., groceries stored in a smart fridge, clothes washed in a public laundry, or goods returned to a department store for exchange).

While the general idea of the kill-feature at first looks much simpler, it also requires a substantial management overhead due to its password-protection requirement (i.e., preventing unwanted silencing of tags, e.g., in a supermarket) that will most likely be impractical in many situations. A manual removal of the tag, e.g., by placing it on a removable label, is much simpler to implement and substantially more user-friendly, as it does not require specialized hardware and can be visually verified. This, however, prohibits value-added services after checkout.

Marc Langheinrich received a master's degree in computer science from the University of Bielefeld, Germany, in 1997. He spent a year as a Fulbright Scholar at the University of Washington in Seattle, USA, and two years as a researcher at NEC Research in Tokyo, Japan. Since October 1999 he is a research assistant in the Institute for Pervasive Computing at the ETH Zurich, Switzerland, where he investigates the intersection of privacy and ubiquitous computing, both from a technical and social perspective. Marc is one of the authors of P3P, an emerging standard for exchanging privacy policies on the Web.

[1] This work is based on an earlier article (in German) by the same author: Marc Langheinrich „Die Privatsphäre im Ubiquitous Computing – Datenschutzaspekte der RFID-Technologie.“ In: Elgar Fleisch, Friedemann Mattern (Eds.): *Das Internet der Dinge – Ubiquitous Computing und RFID in der Praxis*, Springer-Verlag, 2005.

[2] The Auto-ID center was founded in 1999 to develop both RFID tags and standards for identifying everyday things, especially in the supply chain.

[3] Since the Auto-ID center's scheduled close in October 2003, the commercialization and further development of the Auto-ID technology is done by EPCglobal – a joint venture between the Uniform Code Council and EAN International.

[4] A good example are today's RFID-based, contactless ski passes: In order to prevent readers from picking up the pass of someone further down behind, the reading distances must be kept rather short. This inevitably forces skiers to rub their jackets containing their passes in a number of different positions against the reader until the RFID-tag is properly detected by the gate.

[5] As humans contain 45-60% of water, the presence of a only single user already "interferes" with the RFID-system.

[6] A collision only occurs if two tags send a different bit value.

[7] As an example, consider the three tags 00101, 00001 and 00110. The only reader commands an attacker would hear would be: GetNext, GetNext, GetNext (Collision between Tag1, Tag3, and Tag2), Select(1) (Collision between Tag1 and Tag3), Select(0) (Tag1 identified), Select(1) (Tag3 identified), Select(0), GetNext (Tag2 identified).

[8] This obviously only holds for large tag populations, as otherwise the overhead of reading out the EPC separately is too large.

[9] Even an address space of only 64 bits would keep a reader capable of reading 100’000 tags per second busy for over four billion years.