Hiding in Groups: On the Expressiveness of Privacy Distributions

Karsten Nohl and David Evans

University of Virginia
Computer Science Department
{nohl,evans}@cs.virginia.edu

Abstract. Many applications inherently disclose information because perfect privacy protection is prohibitively expensive. RFID tags, for example, cannot be equipped with the cryptographic primitives needed to completely shield their information from unauthorized reads. All known privacy protocols that scale to the anticipated sizes of RFID systems achieve at most modest levels of protection. Previous analyses found the protocols to have weak privacy, but relied on simplifying attacker models and did not provide insights into how to improve privacy. We introduce a new general way to model privacy through probability distributions, that capture how much information is leaked by different users of a system. We use this metric to examine information leakage for an RFID tag from the a scalable privacy protocol and from a timing side channel that is observable through the tag's random number generator. To increase the privacy of the protocol, we combine our results with a new model for rational attackers to derive the overall value of an attack. This attacker model is also based on distributions and integrates seamlessly into our framework for information leakage. Our analysis points to a new parameterization for the privacy protocol that significantly improves privacy by decreasing the expected attack value while maintaining reasonable scalability at acceptable cost.

1 Introduction

RFID labels in consumer products promise a world of new, convenient applications such as smart homes and automated checkout, but also raise serious privacy concerns. Among the many privacy intruding uses of RFID technology are corporate spying and customer profiling. The profiles encode information similar to Internet traces and, hence, have a certain monetary value.

Privacy protocols can protect a tag's identity from an attacker, but incur extra cost that grows with the degree of privacy; the cost becomes prohibitive for perfect privacy. Thus, practical protocols must trade some privacy for lower cost and higher scalability. Previous analyses of scalable protocols concluded that the privacy loss is high [5, 9]. In these analyses, the attacker is not assumed to be rational. Furthermore, although these analyses reveal a lack of privacy, they do not provide practical insights into how to improve the protocols.

We present a new way of measuring privacy that not only captures the privacy of the whole system but also the variance in privacy experienced by different users. Unlike

previous approaches that represent privacy in a single value (e.g., the average group size), our analysis measures privacy loss as a distribution of how much information is leaked by different tags. We use our new privacy metric to derive distributions of information leakage for two example cases: an RFID random number generator that encodes a timing side channel and the tree protocol. Our metric works equally well in modeling these two very different sources of information and can further be used to model almost any source of deterministic or probabilistic information [10].

To derive a better understanding about the economics of privacy attacks and how to improve protection against them, we model a realistic attacker who attempts to collect traces because of their potential financial value. Our attacker is modeled as a function that maps traces to their value. We derive an upper bound on the shape of this function that allows us to model the most capable, yet rational attacker. Analyzing the privacy distribution of the tree-based hash protocol in light of this rational attacker leads to an adjustment of the tree parameters that provides substantially more privacy while incurring no extra tag cost and reasonable additional reader cost. Restructuring the tree improves privacy significantly while preserving scalability.

Our main contribution is a new way of representing privacy in form of a probability distributions which we demonstrate on a timing side channel in Section 3 and on a privacy protocol in Section 4. We then use this metric to analyze the value that an attack has to a rational attacker (Section 5), and propose a simple way for adjusting the tree protocol to better trade-off attack value and protection cost (Section 6).

2 Background

This section provides background information and describes previous work on defining privacy and measuring the privacy properties of RFID systems.

2.1 RFID Systems

The RFID tags we consider are small, cheap, passive radio-readable labels. The tags have unique identification numbers that a reader can read from the tag. The reader uses this ID to look up information from a back-end database.

The basic tags in circulation today provide no privacy protection; any reader can read the unique ID of the tag and look up information from often-public databases. A first step towards more privacy is to restrict access to these databases, but even a random but static identifier on the cards can be used to track people and therefore compromises privacy. Adding privacy protection to tags leads to higher per-tag costs and lower reading ranges (due to the increased power consumption), as well as increased computational cost in the backend system. To support RFID systems with billions of tags, this backend cost must grow sub-linearly with the size of the system.

2.2 Privacy Protocols

Several RFID privacy protocols have been proposed, all of which sacrifice at least one of scalability, availability, or strong privacy. The basic hash protocol, in which a tag

hashes a random nonce with a secret key, provides strong privacy but does not scale well [16]. The database must try the keys of all tags to find the one that matches. This computational overhead is prohibitive for large systems.

A more scalable protocol assigns several secrets to each tag [9]. The secrets are structured in a tree with the tags as the tree leaves. A tag t_i is assigned the secrets $s_{i,1}, s_{i,2}, ...s_{i,d}$ where d is the depth of the tree (all secrets but the last are shared with some of the other tags). When queried, the tag responds with where $H(\cdot,\cdot)$ is a strong one-way function and the r_i values are random nonces. The database executes the basic hash protocol for each tree level to finds the secret used on each level. Once a leaf is reached, the path from the root to the leaf uniquely identifies the tag. In the standard tree protocol, a tree with a constant branching factor at each level is used. This tree-based hash protocol scales well beyond billions of tags. The drawback of the protocol, however, is that secrets are shared among several tags and extracting the secrets from some tags potentially allows tracking others. An attacker can uniquely identify a tag with higher probability when more secrets of that tag are known. We show in Section 6 that increasing the branching factor at the lowest tree level improves privacy and that optimal trees for many scenarios have only two levels. In previous proposals, a binary tree was discussed [9], which according to our analysis has the least privacy of all possible trees.

Buttyan et al. also propose an algorithm for finding the optimal tree of secrets [4]. Their algorithm optimizes for a metric based on the average group size and generally increases the depth of trees. In follow-up work, the same authors and Avoine propose a new protocol that is equivalent to the two-level tree we propose [1]. They show that even in the metric they optimize for, the two-level tree is always superior to the trees their optimization algorithm finds. Our work provides the missing link between measuring privacy and improving privacy within the same framework and explains why trees with fewer levels provide more privacy.

Variations of the tree protocol include the matrix protocol that replicates the same secrets over different tree branches and hence offers less privacy [5]. Other proposed protocols provide strong protection only when rogue reads are rare [15]. If rogue reads occur frequently, these protocols can render the tags disfunctional and leak some information [7].

2.3 Privacy Definition

The highest level of privacy possible in an identity system is *strong privacy* as defined in [7]. Strong privacy requires that an attacker cannot distinguish between a pair of uncompromised tags after interacting with all the tags in the system and extracting secrets from some tags. The only known way to efficiently achieve strong privacy for large-scale systems is by using asymmetric cryptography. Implementing the required public key ciphers on cheap RFIDs, however, is not possible [8].

The basic hash protocol can achieve strong privacy without asymmetric cryptography, but it requires the reader to perform as many cryptographic hashing operations as there are tags in the system. Strong privacy cannot be achieved while also matching the cost and scalability requirements of RFID systems [9, 11].

We define privacy as the state in which no *rational* attacker will attempt to compromise the system. Our definition of privacy acknowledges the fact that some amount of information is always leaked in the real world and that any definition which is too strong cannot be fulfilled. A rational attacker will only attack a system when the expected monetary (or other) return value exceeds the expected cost. Our definition allows for some tags to have relatively weak privacy protection as long as a large majority of tags experience strong protection and the attacker is very unlikely to see many weakly protected tags. In our definition, privacy can be deduced from information leakage, but the exact conversion between the two varies for different attackers and systems. We present a general approach to estimating the value of an attack from a distribution of information leakage in Section 5. Our model is abstract in that we do not assume any specific value of readings, but rather show ways to measure and improve privacy against any rational attacker.

2.4 Measuring Privacy

The privacy of the tree-based hash protocol has been estimated in several research papers. A first analysis calculated the probability that two readings from the same tag can be linked [2]. The paper concluded that the tree-based protocol provides insufficient privacy. We believe that this is too strong a privacy definition, which cannot be achieved, and advocate that the ability of an attacker to build whole traces should instead be considered. An alternative way of calculating the privacy of a system is by measuring the average anonymity of all tags. One such metric measures the average number of tags from which each tag cannot be distinguished [4]. A more precise metric measures the entropy of these groups [11]. Both approaches measure privacy as a single value, which is limiting in two ways. First, condensing privacy into a singular value presumes how the attacker will use the leaked information. Different attackers, however, use the information in diverse ways and hence the privacy of a system depends on the attacker's incentives and capabilities. Secondly, when averaging the information leakage over all tags in a system, information is lost about which parts of the system are mostly responsible for privacy deficits. Understanding the distribution of the information leakage is crucial for reducing the amount of information leaked. In this paper, we use a metric based on Shannon entropy from our previous work that measures the amount of information disclosed by the tags [11]. We extend this metric to consider the distribution of information leaked by tag groups. The tags fall in different groups that can be distinguished while tags within each group cannot. In a group of size g in a system with N tags, $\log_2(N/g)$ bits of information can be learned from each tag.

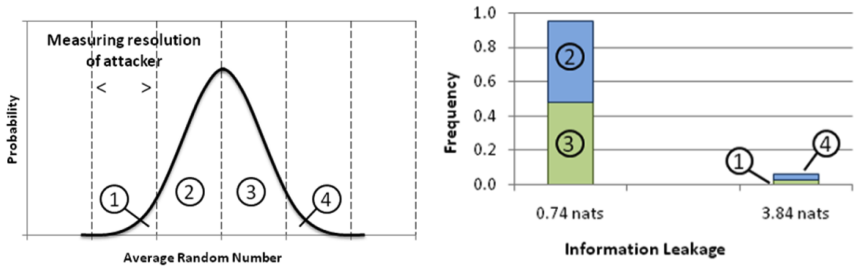
3 Side Channel Information Leakage

Privacy is potentially compromised at many layers including side channels. Side Channels are often caused by physical variance across different tags and can be observed as different timing, power consumption, or antenna characteristics. We are analyzing the varying lag between the moment a tag is supplied with power and when it starts operating. Because this latter time cannot be observed directly, we measure it indirectly

through the tag's choice of random number that is used during anti-collision. These random numbers are required to match certain statistical randomness properties by current RFID standards. The EPC standard, for example, requires all numbers to be generated with approximately the same frequency; every number must be generated with a frequency of 80%-125% the average frequency [6]. Note, that these requirements do not guarantee randomness in a cryptographic sense. In particular, an attacker can often bias the distribution of values, which, besides breaking anti-collision, potentially compromises privacy. Random number generators found on current tags (including cryptographic tags) generate random numbers using a linear feedback shift register (LFSR) with constant initial condition [12]. Each random value, therefore, only depends on the number of clock cycles elapsed between the time the tag is powered up (and the register starts shifting) and the time the random number is extracted. The variation in the choice of random number is hence a timing side channel that allows different groups of tags to be distinguished.

One type of commonly found tags, for instance, generates 16-bit random numbers using an LFSR of the form $x^{16}+x^{14}+x^{13}+x^{11}+1$ [12]. The register is clocked at 106 kHz and wraps around every 0.6 seconds after generating all 216 possible output values. The numbers are, therefore, only unpredictable if an attacker cannot measure or control the timing with accuracy smaller than 0.6 seconds. Realistic attackers will more likely measure with micro- or nano-second accuracy. Since the generated number only depends on timing, which is controlled by the reader, an attacker has a large degree of control over which number will be generated. Using custom-built reader firmware, we were able to make tags generate the same "random" number repeatedly by querying the tag at exactly the same time after switching on the reader field. In theory, the numbers generated for a given timing should be the same for all tags, because the circuitry that generates the numbers is the same and no physical randomness is used. Therefore, no information about the tag should be learned from the choice of number. In practice, however, we observe that due to manufacturing differences, groups of tags can be distinguished based on how quickly they start operating after the reader field is switched on. The distribution of process variance follows a typical normal (Gaussian) distribution and so does the average expected value generated by different tags. Most tags have too little variance to be distinguishable while few tags power up sufficiently slower or faster than the average so that the expected random value is slightly before or after the average value in the LFSR sequence. Figure 1a shows a typical distribution of expected average values for different tags that we estimated from our experiment. In this experiment we queried several different cards for a random number after exactly the same time. The average of each tag is slightly biased from the average of all tags.

The number and size of the groups that can are distinguishable due to their random numbers depends on the amount of process variation and the measuring accuracy of the attacker. The more variation exists among the tags and the better the attacker can control the timing of the tag, the more groups can be distinguished. For simplicity of our example, we assume the expected values to follow a normal distribution with standard deviation σ and an attacker with a timing resolution of 2σ. Virtually all tags can be placed in one of four groups as shown in Figure 1: Those having expected values slightly smaller or larger than the average, or significantly smaller or larger than the average.



(a) (b)

Fig. 1. Information disclosure of weak random number generator (a) distribution of varying expected value due to process variation; (b) distribution of information leakage.

An attacker learns more information from tags in smaller groups. A group of z tags corresponds to $\log_2(N/z)$ bits of leaked information where N is the total number of tags that an attacker potentially encounters (and is the total number of tags in the system unless the attacker has extra knowledge to exclude some tags). For the convenience of the following calculations, we calculate information in *nats* rather than *bits*. Nats are similar to bits but computed with base e; 1 nat equals $1/\ln(2)$ bits (approx. 1.33 bits). A group size z therefore corresponds to $\ln(N/z)$ nats of information. The distribution of information leakage for tags equipped with the weak random number generator is shown in Figure 1b.

In our example, each of the two groups around the average value holds 47.7% of the tags. The information leakage from tags in these two large groups is $\ln(1/0.477) = 0.74\text{nats} = 0.98\text{bits}$. The leakage from tags in the two smaller groups that divert more from the average value is consequently much higher at 3.84 nats (5.10 bits).

The privacy of a system is directly related to the distribution of information leakage. For the analyzed random number generator, privacy is therefore also directly related to the amount of process variation. Privacy can be significantly increased by lowering the process variation or by excluding a small number of outliers from the system. A better lesson still to be learned from this analysis is that any RNG design that gives control over the generated numbers to the attacker is clearly flawed, especially when also used for purposes other than anti-collision.

4 Tree Protocol Information Leakage

In the same way that we found the distribution of information leakage for the number generator, we can find similar distributions for most other sources. To analyze the privacy of the tree protocol, we need to know the distribution of group sizes and the likelihood that a randomly chosen tag falls into a group of a certain size. Tags are indistinguishable to the attacker if they use the same subset of the secrets known to the attacker. The larger the group of indistinguishable tags is, the more privacy is provided for the tags in the group. The distribution of group sizes depends on the tree parameterization and the set of secrets known to the attacker.

The tree protocol provides strong privacy if none of the secrets are known to the attacker: all N tags in circulation are in one large group of size N. As the attacker learns secrets from the tree, however, smaller groups of tags can be distinguished and strong privacy is lost. When considering a tree with a depth of D and a spreading factor of k, the first broken tag gives the attacker d secrets, one on each tree level. These secrets help to distinguish the broken tag from its $(k-1)$ immediate neighbors, from their (k^2-k) immediate neighbors, and so on. For example, Figure 2 depicts a tree of size $N = 256$ with spreading factor $k = 4$. After the bottom left tag is compromised, an attacker can group the tags into groups of sizes 3, 12, 48, and 192 tags.

For $z = 3, 12, 48, 192$ there are z tags in a group of size z and z/k tags in smaller groups. In our example, there are 48 tags in a group of size 48 and 12+3+1 tags in smaller groups. For all other values of z there are less than $z+z/k$ tags in groups smaller or equal to z. Therefore, the probability that a randomly chosen tag falls into a group of size z or smaller after the secrets on a single tag have been captured is upper-bounded by

$$\Pr(Z \leq z) \leq \frac{k}{k-1} \cdot \frac{z}{N}. \tag{1}$$

The one broken tag is no longer considered part of the system and (1) is defined over the range of group sizes actually found in the tree; that is:

$$k-1 \leq z \leq N \cdot \frac{k-1}{k}.$$

To simplify the following calculations, we consider only the upper bound of (1). This bound corresponds to the case where one group for each possible size $(1, 2, 3, \ldots)$ exists. These groups each hold $k/(k-1)$ tags. While this case is impossible to achieve in reality because a group of size z should have z members, it provides a close enough upper bound on the real distributions of groups.

This upper bound on the cumulative distribution is shown in Figure 3a along with the values of the real distribution. Note that the upper bound diverges most from the real distribution for those groups that leak the least information, and least for the more important groups that leak the most information. In our example tree of 256 tags with one broken tag, there are 4/(4-1) = 1.33 tags in a group of size one, another 1.33 tags in a group of size 2, and so forth. This configuration never appears in reality but provides a close upper bound on the real distribution; and unlike the real values, this bound can be expressed in a closed-form probability distribution.

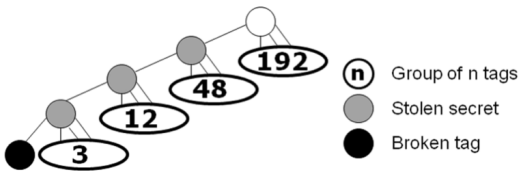
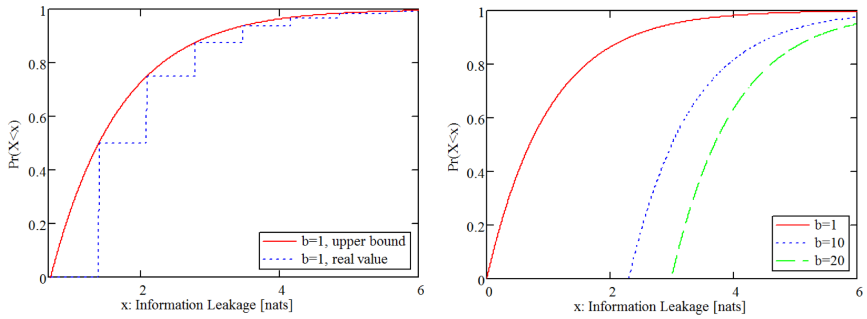


Fig. 2. Groups of tags that an attacker can distinguish after one tag compromise.



(a) (b)

Fig. 3. Probability that for a tree with b broken tags less than x nats of information are leaked by a randomly selected tag, k is large. (a) real value vs. upper bound, one broken tag; (b) upper bound for different numbers of broken tags.

The probability that x or more nats are leaked (as defined in Section 3) by a randomly chosen tag is upper bounded by:

$$\Pr(X \geq x) \leq \frac{k}{k-1} \cdot \frac{1}{e^x}.$$

This is defined over the range of inputs that correspond to group sizes actually found in the tree; that is:

$$\ln\left(\frac{k}{k-1}\right) \leq x \leq \ln\left(\frac{N}{k-1}\right).$$

So far, we only considered the case of a single broken tag. Assuming tags are evenly distributed, each additional broken tag adds the same number of members to each group (with the exception of the largest group which shrinks in size). For b broken tags[1], the probability that at least x nats of information for a given tag are disclosed becomes

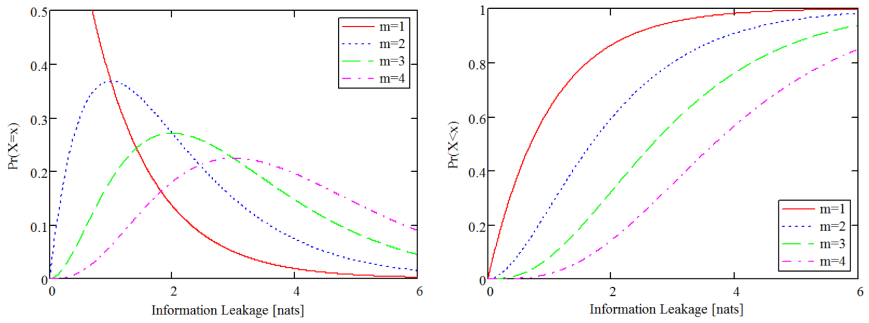
$$\Pr(X \geq x) \leq b \cdot \frac{k}{k-1} \cdot \frac{1}{e^x} \tag{2}$$

for the range

$$\ln\left(\frac{k}{k-b}\right) \leq x \leq \ln\left(\frac{N}{k-1}\right).$$

The probability that the system defeats an attacker who requires at least x nats of information to be successful is: $\Pr(X < x) = 1 - \Pr(X \geq x)$. Figure 3b shows this probability for different numbers of broken tags. Note that the distribution is independent of the number of tags in the system and for large trees it is essentially independent of the spreading factor, k. A realistic attacker will certainly behave more complex than this simple threshold. We address this concern in the Section 5 with an extended model that acknowledges the rational and adaptive behavior of realistic attackers.

[1] Our approximation is closest if b<k, but still valid otherwise.



(a) (b)

Fig. 4. (a) Distribution of information leaked by collection of m tags. (b) Probability that information leaked by m tags is smaller than x nats.

Multi-Tag Attack. A sophisticated attacker will use all available information to distinguish individuals. In particular, all tags that a person carries will be used to track that person. In previous work, we showed how such a multi-tag attack can be used to distinguish individuals in large systems despite protection measures [11]; here we extend those results using distributions of information leakage.

Suppose each individual carries m tags that are randomly selected from the tree. To create an identifier for a person, the attacker simply concatenates the information learned from the various tags the person carries. The different identifier that can be build this way separate all individuals into separate groups, which can again be reflected by a distribution of information leakage. The information learned from each of the tags follows the exponential distribution described by (2). Summing several exponential distributions leads to a gamma distribution. Hence, the probability that a total of x nats of information are leaked from a person that carries m tags is:

$$\Pr(X = x) = \frac{x^{m-1}}{(m-1)!} \cdot b \cdot \frac{k}{k-1} \cdot \frac{1}{e^x}.$$

This distribution is shown in Figure 4a for different m. The probability that not more than c nats of information are leaked from a randomly chosen tag can be calculated as the integral of this function up to c, which is shown in Figure 4b. Note that the graph shows the probability that the system is *not* compromised, so lower values indicate a higher probability of privacy compromise.

5 Distinguishing Traces

The information learned through RFID traces can be used in many different applications where profiling, tracking, or monitoring of goods or individuals is required. Privacy-intruding uses of RFIDs include surveillance of individuals, corporate espionage, and profiling of consumers.

Rogue customer profiling is the most frequently discussed scenario in the context of RFID privacy. Traces collected by a rogue reader could be used in ways similar to Internet traces to build customer profiles and enable price discrimination [13], and therefore constitute a certain value. In many scenarios, RFIDs primary purpose is to build such profiles and customers are often provided with price incentives to participate in loyalty schemes. But while legitimate collecting of consumer data is transparent and provides incentives to the customer, rogue readers will try to read the same information from the tags without owner consent. Another rational attack scenario is corporate espionage where information is collected from RFID labels on products to learn their internal business information of competitors. Lastly, tracking attackers keep individuals under surveillance through RFID readings. Our analysis of the expected attack value in this section and the proposed modification of the tree protocol in Section 6 apply equally to all three types of attacker. In this analysis, we avoid making restrictive assumptions about the actual use of collected traces or attempt to quantify their value. Instead, we assume that in any attack, the attackers' objective is to distinguish tags in order to build traces, which is more likely when the attacker has more information about the tags. Our privacy metric, therefore, measures the amount of information the attacker learns about different tags. The magnitude of the expected attack value and the effectiveness of our proposed defense will vary among the different attackers and depend on the expected pay-off and the cost of substitute tracking techniques. Privacy is consequently achieved when the expected cost of any attack exceeds the expected return. As long as information from RFID traces is an essential source in the attack, however, our analysis covers a substantial portion of the attack economics and our protocol improvement significantly improves privacy.

Other Information Sources. Attackers are not limited to information from the tree protocol or random number generator. Our approach of modeling information leakage as a probability distribution applies just as well to all other information sources such as physical side channels. Sources that might be used in an attack include the physical characteristics of the tag (e.g., radio fingerprint), meta-information (e.g., location and time of read), and information from other detection systems, such as biometric identification systems like face and voice recognition. The exact distribution of many of these sources is as of yet unknown. In the next section we show how all these additional sources can be modeled as attacker strategy.

Threat Model. We are considering an attacker that mines RFID data sets for profitable traces, where a *trace* is a set of readings from the same tag. In order to build traces, an attacker wants to link the different RFID readings collected from the same individual. Each reading consists of time, place, the randomized tag identifier, and potentially further metadata. The attacker's goal is to extract individual traces form a collection of many intermingled traces. The likelihood that the attacker will be successful grows with the amount of information leaked by the tags the individual carries. Readings have higher value to the attacker when they carry more information. The exact value function that maps information to value is unknown and will be different for different attackers. We can, however, model the approximate shape of such a function and find an upper bound that corresponds to the strongest attacker that is most skilled in using additional information to distinguish traces.

An attacker wants to extract information from data collected by reading RFID tags. The data set the attacker collects is composed of many intermingled traces from different tags. We assume that each individual trace becomes valuable only when it can be separated from all other traces thus identifying a set of readings from a single tag (or a conjoined group of tags). Whether a trace can be separated using data from the protocol level depends on the secrets known to the attacker. The previous section derives the expected sizes of tag groups distinguishable on the protocol level for an attacker with a given number of compromised tags. To separate those traces that are indistinguishable at the protocol level, the attacker will further employ data mining techniques, use additional information sources such as the weak random numbers from Section 4, other side channel information, or contextual information such as place and time of read to distinguish traces. To capture the success of the attacker in doing so, we introduce two functions: the *attacker strategy function*, which describes the sophistication of the attack, and the *binning function*, which captures the clustering of traces.

The attacker strategy function encodes the probability with which an attacker can distinguish traces that are indistinguishable at the protocol level. The function captures the side channel information and data mining techniques that are available to the attacker, and varies widely for different attackers. Even though the function is generally unknown—even to the attacker—we can derive an upper bound on it.

First, we define P as the average probability that a set containing two traces can be distinguished. This average probability is closely related to the success of our attacker who is interested in collecting a large number of traces with high probability. Other approaches to modeling the privacy of RFID systems have only considered the probability that the privacy of any of the tags in a data set can be compromised [3]. These analyses, hence, measure the least privacy experienced by any of the tags rather than the average, and consequently conclude that sufficient protection cannot be achieved for all users of a system. In contrast, privacy in our model is achieved if the average protection is high enough to discourage all rational attackers from attacking a system while the actual protection experienced by different users can vary.

Given a set of any number of readings known to come from two different tags, the attacker can separate the readings into two groups based on the tag from which they were generated with probability P. The actual probability also depends on the number of readings for each tag in the set. The attacker, however, does not control the number of readings collected form a specific tag in different locations, but only the total number of readings collected from all tags. Hence, our definition of P does not depend on the number of readings, just on the number of different tags which could have generated the readings. This models an attacker who tries to extract traces from a data set without having detailed knowledge of the composition of the data.

Given this definition of P for separating two traces, it follows that traces cannot be separated from groups of three intermingled traces with average probability better than P^2. If the chances of extracting a trace from a set of three traces were higher, an attacker could distinguish two traces with a probability higher than P by intermingling an additional trace. We can generalize to any number of intermingled traces:

Trace Extraction Theorem: *Let the average probability over all traces that two traces can be distinguished be P. Then a trace cannot be extracted from a set with $g > 1$ traces with an average probability better than $P^{(g-1)}$.*

To prove this theorem we show that when a trace is added to a set of traces, extracting that trace from the set is at least as difficult as distinguishing the trace from every member of the set. We further show that this probability is always maximized when, given the average probability of distinguishing two traces, P, all pairs of traces can be distinguished with the same probability. The proof is given in the Appendix.

Hence, the strategy function is upper-bounded by:

$$S(g) = P^{(g-1)}.$$

Second, the binning function describes the distribution of traces into the different groups that can be distinguished in the tree using protocol information leakage. Given a distribution of groups as derived in Section 4, a system with size N, and a data set with $t+1$ intermingled traces, the probability that at most g traces from a group of size z are included in the data set is:

$$B(z,g) \leq \binom{t}{g} \cdot \left(1 - \frac{z}{N}\right)^{t-g}.$$

Expressed in terms of encoded information, the probability that g indistinguishable traces with x nats of entropy (that is, traces from a group with size $z = N/e^x$) are found in the set is:

$$B(x,g) = \binom{t}{g} \cdot \left(1 - \frac{1}{e^x}\right)^{t-g} \cdot \left(\frac{1}{e^x}\right)^{g}.$$

The binning function is shown in Figure 5a for traces with different entropies.

Last, the *success function* encodes the probability that a trace with given entropy can be extracted. This is the likelihood that the trace is intermingled with $(g-1)$ other traces, $B(x,g)$, multiplied by the probability that the attacker can extract a trace from a set of g traces, $S(g)$, and summed over all possible mixes ($g = 0, 1, ..., t$; where $g = 0$ is a single, not intermingled trace):

$$Su(x) = \sum_{g=0}^{t} \left(B(x,g) \cdot S(g)\right).$$

The success function is shown in Figure 5b. To determine whether an attack is successful, the success function needs to be interpreted in light of the attacker's requirements. The value of an attack to the attacker depends on the likelihood that traces can be extracted from the collected data, which is encoded in the success function. Multiplying the success function with the likelihood that a trace with given entropy occurs (from Section 4), and integrating over all possible entropies (that is, the entropies from that largest to the smallest group) gives us the expected value of the attack:

$$Value = \int_{ent(lrg)}^{ent(sm)} \left(Su(x) \cdot \Pr(X = x)\right) dx \tag{3}$$

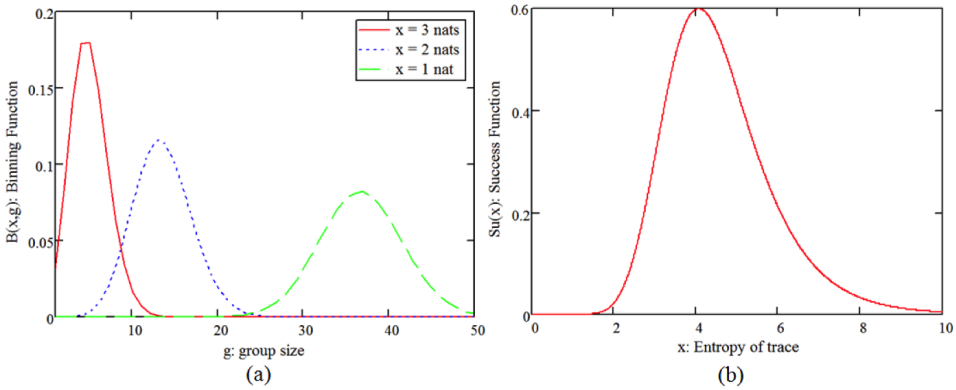


Fig. 5. Binning Function: probability that a trace with x nats of entropy is intermingled with g other traces; data set with $t = 100$ traces. (b)Success Function: probability that a trace with x nats of entropy can be extracted; $t = 100$.

where $ent(lrg)$ and $ent(sm)$ are the entropies of the largest and smallest groups in the system. As shown in Figure 4b, only certain group sizes have a high likelihood to contain useable information. In both the random number generator and the tree protocol, this is due to the fact that a large majority of the tags hides in few large groups, but only the few tags in small groups can easily be distinguished. Hence, the majority of the attack value is generated by these groups of the smaller sizes while the majority of tags contributes only very little. In the next section, we use this insight to design a protocol modification that decreases the attack value and defeats a rational attacker.

Most information sources are fuzzy; items with RFID tags get swapped, physical characteristics often depend on the environment and face recognition produces false matches. Further research is needed to understand how data mining techniques can be used to overcome this intrinsic noise and refine the true information. The attack value will depend on how well an attacker can do this.

Regardless of which other information sources are integrated into a large-scale privacy attack, the RFID data will be an essential source in any such attack unless the tags are better protected. Limiting the leakage from RFID tags can significantly decrease the attack value, even when all the other (unknown) information sources are left unchanged. Hence, we focus next on limiting the information leaked from this protocol and propose a simple modification that significantly decreases the generated expected attack value.

6 Minimizing Attack Value

We assume rational attackers have financial goals. Such an attacker will only attack a system if the expected value of the traces acquired exceeds the cost of the attack. The value of an attack is given by (3). The insight that the bulk of the attack value is generated by the tags in smaller groups points to a simple but very effective improvement of the tree protocol: restructure the tree so that no tags fall in groups smaller than some

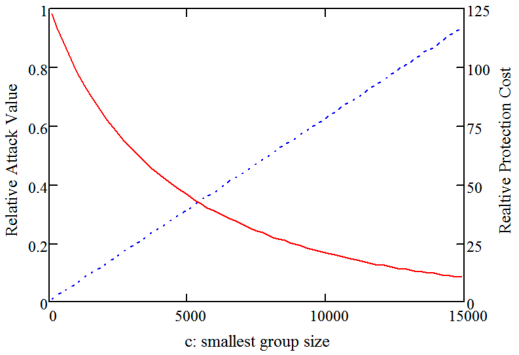


Fig. 6. Trade-off between attacker value and protection cost. Scenario of a large system with many broken tags and many tags per person, specifically $N = 10^7$, $d = 4$, $b = 50$, $m = 10$, $P = \frac{1}{2}$, and $t = 100$.

threshold. Moving tags from these smallest groups to larger groups decreases the attack value significantly.The most scalable tree for a given tree depth, d is a balanced, fixed-k tree that has a constant spreading factor of $k = N^{1/d}$. We assume that d is fixed because it is limited by the tag memory and by the maximum reading time. Varying the spreading factor for the whole tree does not provide an effective trade-off. Varying the spreading factor only at the last level of the tree, however, preserves most of the scalability, while significantly improving privacy.

The privacy-improved tree is constructed by dividing the tags into groups of some threshold size c and constructing a balanced, fixed-k tree with these groups as the leaves. The threshold is chosen to be larger than the largest group size that still carries a high value to the attacker. The computational time required to find a tag is the time to find the correct leaf plus the time to identify the correct tag within that leaf:

$$Cost = \left(\frac{N}{c}\right)^{\frac{1}{d-1}} \cdot (d-1) + c.$$

In comparison, the balanced tree has a maximal cost of $N^{1/d}d$. Increasing c increases privacy and computational cost. The attack value of the modified tree is computed using (3) but with a larger minimum group size. The attack value is:

$$V_{all}(c) = \int_{\ln\left(\frac{k}{k-b}\right)}^{\ln\left(\frac{N}{c-1}\right)} V_{gen}.$$

A fixed-k tree for this setup requires a spreading factor of 56. Increasing the size of the groups on the last level beyond 56 decreases the attack value. It falls to 40% of the maximum value at a group size of 1100 tags while increasing the reader cost 9 times. Limiting the attack value to 20% leads to a 30-fold cost increase. These correspond to 1139 and 3729 hashing operations, which is still orders of magnitude below the 10^7 operations required to reduce the attack value to zero using the basic hashing scheme with no shared keys.

For a large group of attackers and for virtually all RFID systems, minimal group sizes of $\sqrt{N}$ will provide sufficient privacy. The resulting tree has only two levels, which requires smaller memories on the tag and leads to quicker reading times when compared to a deeper tree. The computational cost is still not prohibitive even for very large systems. In a system with one billion tags, each read requires about 6000 hashing operations, which takes only a fraction of a second on a general-purpose processor.

The attacker value can be further reduced by decreasing k on the first level and increasing it on the second level. The additional computational overhead could be offset by using custom designed hardware. State of the art implementations of standard hash functions provide a throughput of hundreds of Mbit/s [14], which equals millions of hashing operations per second. A hashing computer build from many of these chips can execute on the order of a billion hashing operations per second at reasonable cost. For the system with a billion tags, this would support authenticating more than 100,000 tags per second on a single server. The appropriate tree for many RFID privacy scenarios therefore has only two levels and is hence the opposite configuration from the initially proposed binary tree [9].

7 Conclusions

Modeling information disclosure as a probability distribution of leaked information exposes the parts of a system responsible for most of the privacy lost. Analyzing secret-trees and side channels using such distributions helps to identify a small subset of tags as the source of most of the privacy loss and provides new insights into the trade-off between cost and privacy. The privacy distribution of secret trees and many other sources can be approximated by an exponential distribution. The distribution for the tree protocol depends on the number of broken tags, but not on the system size or the spreading factor. Good designs can hence be found that apply to a wide range of applications. When information from several tags or other sources is combined by an attacker, the overall information leakage can be modeled using a single gamma distribution. Our approach of expressing all privacy leaks in the form of probability distributions enables designers of privacy protection to identify the weakest link and thereby estimate the privacy of the overall system, which has not previously been possible.

When combined with a rational attacker model, identifying the weakest part of the tree protocol enabled us to find new parameters for the tree with much improved privacy. Our attacker model takes into account the value of different traces and assumes that an attack is successful only if the overall value exceeds the attack cost. Attackers can be modeled by a function that assigns values to different amounts of information leakage. Without making restrictive assumptions on the actual incentives of the attacker we can prove an upper bound on the value function that corresponds to the most capable attacker. This overall value is mostly generated by the tags in the smallest groups at the last level of the tree. Varying the size of these groups trades increased computational cost for decreased attack value. Lowering the attack value by up to 80% incurs no cost on the tag and only a small overhead in the backend. Even though our parameterization might seem obvious in retrospect, it was not stated prior to our analysis. In fact, the previously proposed binary tree appears to provide the least privacy of all possible setups.

This underlines that good models for information leakage and a good understanding of the attacker's incentives are needed when designing new privacy protocols.

Acknowledgment. This work has been partly supported by the National Science Foundation through Award 0541123.

References

1. Avoine, G., Buttyan, L., Holczer, T. and Vajda, I. Group-Based Private Authentication International. *Workshop on Trust, Security, and Privacy for Ubiquitous Computing*, 2007.
2. Avoine, G., Dysli, E. and Oechslin, P. Reducing Time Complexity in RFID Systems. *Selected Areas in Cryptography*, 2005.
3. Avoine, G. and Oechslin, P. RFID Traceability: A Multilayer Problem. *Financial Cryptography*, 2005.
4. Buttyan, L., Holczer, T. and Vajda, I. Optimal Key-Trees for Tree-Based Private Authentication. *Workshop on Privacy Enhancing Technologies*, 2006.
5. Damgard, I. and Ostergaard, M. RFID Security: Tradeoffs between Security and Efficiency. *Cryptology ePrint Archive*, 2006.
6. EPCGlobal Inc. Class 1 Generation 2 UHF Air Interface Protocol Standard v1.0.9, 2005.
7. Juels, A. and Weis, S. Defining Strong Privacy for RFID. *Cryptology ePrint Archive*, 2006.
8. Kumar, S. and Paar, C. Are standards compliant elliptic curve cryptosystems feasible on RFID? *Workshop on RFID Security*, 2006.
9. Molnar, D. and Wagner, D. Privacy and Security in Library RFID: Issues, Practices, and Architectures. *ACM CCS*, 2004.
10. Nohl, K. and Evans, D. Quantifying Information Leakage in Tree-Based Hash Protocols. *University of Virginia, CS Dept., Technical Report UVA-CS-2006-20*, 2006.
11. Nohl, K. and Evans, D. Quantifying Information Leakage in Tree-Based Hash Protocols. *Conference on Information and Communications Security*, 2006.
12. Nohl, K., Evans, D., Starbug and Ploetz, H. Reverse-Engineering a Cryptographic RFID Tag. *USENIX Security*, 2008.
13. Odlyzko, A. Privacy, Economics, and Price Discrimination on the Internet. *International Conference on Electronic Commerce*, 2003.
14. Satoh, A. and Inoue, T. ASIC-Hardware-Focused Comparison for Hash Functions MD5, RIPEMD-160, and SHS. *International Symposium on Information Technology*, 2005.
15. Tsudik, G. YA-TRAP: Yet Another Trivial RFID Authentication Protocol. *International Conference on Pervasive Computing and Communications*, 2006.
16. Weis, S., Sarma, S., Rivest, R. and Engels, D. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. *International Conference on Security in Pervasive Computing*, 2003.

Appendix: Trace Extraction Theorem Proof

A *trace* is a series of readings of the same RFID tag, where each reading is a data item consisting of time, place, a randomized identifier, and potentially further metadata. Traces are denoted t_i. A *set of traces*, $T = \{t_i, t_j\}$, is an intermingled collection of traces from different tags that are indistinguishable on the protocol level. The event that trace t_i can be extracted from set $\{t_i, t_j\}$ is denoted $\{t_i, t_j\} \rightarrow t_i$. The probability of this event, $\Pr[\{t_i, t_j\} \rightarrow t_i]$, will be denoted $p_{i,j}$. It follows that $p_{i,j} = 1$ and $p_{i,j} = p_{j,i}$.

Denote by S_k the set of all sets of k traces. $n = |S_1|$ is the number of traces in the system. The average probability that any two traces can be distinguished, denoted as P, is:

$$P = \frac{1}{\binom{n}{2}} \cdot \sum_{i,j,i<j} p_{ij}$$

We assume that P is known and show how the maximum average probability of distinguishing more than two intermingled traces depends on P.

Lemma: *If a trace t_i cannot be extracted from $\{t_i, t_j\}$ then the trace cannot be extracted from $\{t_i, t_j, t_z\}$ for any z.*

Proof: Suppose t_i was extractable from $\{t_i, t_j, t_z\}$ but not from $\{t_i, t_j\}$. By adding t_z to $\{t_i, t_j\}$, t_i becomes extractable from $\{t_i, t_j\}$.

Corollary: *Trace t_i can be extracted from set T only if it can be distinguished from each member of T, hence*:

$$\Pr[T \rightarrow t_i] \leq \prod_{t_j \in T} p_{i,j}$$

Trace Extraction Theorem: *Let P be the average probability that two traces can be distinguished. A trace cannot be extracted from a set with $g > 1$ traces with an average probability better than $P^{(g-1)}$.*

Proof: The value $\Pr(g)$ is the average probability of extracting any member from any set of g traces:

$$\Pr(g) = \frac{1}{k \cdot \binom{n}{k}} \cdot \sum_{G \in S_k} \sum_{b \in G} \Pr[G \rightarrow b]$$

Following the Corollary, this is at most:

$$\Pr(g) = \frac{1}{k \cdot \binom{n}{k}} \cdot \sum_{a \in S_1} \sum_{G \in (S_{k-1}/a)} \prod_{b \in G} p_{a,b}$$

Assume towards contradiction that $\Pr(g)$ was maximized for a distribution where not all $p_{i,j}$ values are equal:

$$\exists p_{a,b}, p_{c,d}, \alpha > 0 : p_{a,b} = p_{c,d} + 2 \cdot \alpha$$

Assuming, for now, that all other pairs are in equilibrium, shifting weight from $p_{a,b}$ to $p_{c,d}$ decreases those summands of $\Pr(g)$ that contain only $p_{a,b}$, but increases those summands that only contain $p_{c,d}$ by the same amount. Those summands that contain both pairs increase by a factor of:

$$\frac{\left(p_{a,b}-\alpha\right)\left(p_{c,d}+\alpha\right)}{p_{a,b}p_{c,d}} = 1+\frac{\alpha^2}{p_{a,b}p_{c,d}}$$

For any distribution of the $p_{i,j}$ values, starting at the largest gap and successively shifting weights from larger to smaller values, constantly increases $\Pr(g)$ up until all $p_{i,j}$ are equal. The probability of extracting traces from sets of any size is hence maximized when all $p_{i,j}$ are in equilibrium distribution and is at most:

$$\Pr(g) \leq \prod_{g-1} p_{i,j} = P^{(g-1)}$$