2009 Annual Computer Security Applications Conference

Privacy through Noise: A Design Space for Private Identification

Karsten Nohl
University of Virginia
nohl@cs.virginia.edu

David Evans
University of Virginia
evans@cs.virginia.edu

***Abstract*—To protect privacy in large systems, users should be able to authenticate against a central server without disclosing their identity to others. Private identification protocols based on public key cryptography are computationally expensive and cannot be implemented on small devices like RFID tags. Symmetric key protocols, on the other hand, provide only modest levels of privacy, but can be efficiently executed on servers and cheaply implemented on devices. The privacy of symmetric-key privacy protocols derives from the fact that an attacker only ever knows a small fraction of the keys in a system while the legitimate reader knows all keys. We propose to amplify this gap in the ability to distinguish users by adding noise to user responses. We focus on scenarios where an attacker is not able to acquire multiple different reads known to be from the same device, and justify this threat model by proposing a simple modification to RFID tag designs. In such scenarios, we can use noise to blur the borders between groups of users that the attacker would otherwise be able to distinguish. We evaluate the effectiveness and cost of this randomization and find that the information leakage from the tree protocol can be decreased to two thousandths of its original value with 150 times the number of server-side cryptographic operations and minimal cost to the tag. Degrees of privacy up to those achieved by public key protocols can be reached while staying well below the cost of public key cryptography.**

I. Introduction

The need for an ever-growing number of small devices and tokens to identify or authenticate creates a permanent threat to privacy. To preserve privacy, users need to be able to identify themselves to a legitimate server without disclosing their identity to unauthorized readers. Private identification can be achieved through public key cryptography. Public key encryption, however, cannot be implemented on small devices. Hardware implementations of public key ciphers are at least six orders of magnitude larger or slower than symmetric primitives such as block ciphers, stream ciphers, and hash functions. Radio-readable credit cards are one example of a large-scale system where lack of privacy protection has raised some alarm [7]. The resource constraints of these contactless credit cards and the increasingly large number of issued cards require a privacy solution that scales gracefully and comes at little extra cost per card. However, privacy protocols specifically designed to support the area, power, and scalability constraints of large RFID systems have repeatedly been shown to disclose too much information [2][19][14][9].

We propose a new threat model for private identification systems in which an attacker is not able to interact with a particular tag for a prolonged period of time. This assumption is realistic since it is already assumed that there is no physical security on the tags; an attacker who acquires physical access to a tag can easily extract all key material from the tag. In scenarios where it is safe to assume an attacker cannot knowingly conduct repeated interactions with the same tag, our randomized protocol provides design choices on the trade-off curve between scalability and privacy that lie between current RFID protocols and public key protocols.

Several RFID privacy protocols have been proposed for low-security RFID applications such as retail logistics, where readings are frequent and data has to be available instantly (e.g., [12]). These protocols are cheap for both the tag and the server. Whether private identification protocols need to be particularly inexpensive on the server side is an open point of discussion. Applications such as credit card transactions, for example, already involve extensive server computation for fraud detection. Privacy protocols for these applications can hence be more expensive for the server.

Our protocol improves upon the Molnar-Wagner tree-based RFID privacy protocol in which each user is assigned several secrets, many of which are shared with some of the other users [12]. The tree protocol provides only modest levels of privacy since an attacker who steals secrets from one user can distinguish groups of other users. We improve the privacy of the tree protocol by flipping a random set of bits in user messages. The randomization increases the server cost, but never leads to an incorrect or failed identification since it is not done at the last level of the tree.

The idea of using noise to improve a property of a cryptographic process is inspired by HB protocols that use randomization to make a function non-invertible [8]. In our protocol, we add noise to the output of a one-way function to gain privacy. The ideas are orthogonal and can be combined by using an HB function as the

1063-9527/09 $26.00 © 2009 IEEE
DOI 10.1109/ACSAC.2009.55

one-way function in our protocol.

Contributions. The primary contribution of this paper is the development and analysis of a new way of providing privacy for a class of large-scale, low-cost identification systems. In particular:

- We provide a definition for private identification protocols and introduce a distinguishability game for measuring the privacy of a private identification protocol (Section II-B). We provide a metric for evaluating the privacy of a private identification based on a refinement of a metric that was introduced in [15] (Section II-C).
- We describe a new threat model for limited interaction systems (Section II-A).
- We introduce the randomized tree protocol (Section III), and analyze its privacy properties of the randomized tree protocol (Section III-A). We also consider the privacy properties under a stronger threat model in which an attacker can perform multiple reads with a target tag (Section III-C).
- We present and analyze an alternate version of the protocol that takes advantage of selective randomization (Section III-B).
- We analyze the costs of the protocol, including results from simulations and a closed-form approximation of the server workload (Section IV).

II. Private Identification

In various applications, including radio-enabled credit cards, protocols are needed to protect the user's privacy. These protocols must be efficiently implementable on identification tokens and must not lead to prohibitive computational overhead for the back-end server.

A. Threat Model

We assume an attacker who is motivated to track individuals based on the tagged items they carry. The attacker's goal is to distinguish between tags as accurately as possible to build a profile of individual users. Attackers do not necessarily need to uniquely identify a tag to obtain information useful in building a user profile since they can combine protocol-level information with contextual information and combine data from multiple tags carried by the same user. Hence, an attacker can succeed by distinguishing tags in small groups rather than by uniquely identifying tags.

The attacker can set up rogue readers at various locations such as doorways, and can carry a portable rogue reader in a backpack. In addition, we assume attackers can acquire many tags and acquire the secrets stored on those tags. Note that an attacker who can physically plant a tag on an individual can easily track that individual; hence, there is little value in designing a protocol to resist attacks that involve physical access to a tag that returns to the system. We further assume that the time an attacker can surreptitiously interact with a isolated tag is limited. An attacker may be able to interact with a tag for a short while by standing near the user with a backpack, but would not be able to do this for a prolonged period without raising some suspicion. As another example, if privacy paranoid users put their tag-enabled cards in a Faraday-cage wallet the tags are only exposed for the short period of time when they are taken out for use.

Passive RFID tags can easily be designed to impose a time limit between separate readings so that an attacker would need prolonged access to an isolated tag to obtain multiple readings known to come from the same tag. To prevent an attacker from collecting a large number of readings, while maintaining the functionality of the tag, a tag should respond with the same message when queried multiple times in the same location, but should respond with different messages when queried in different locations. This behavior can be achieved for RFID tags by storing the once-computed and randomized hash in capacitor-backed RAM. When the tag has left the reader field for some time, the capacitor is depleted and the stored value is lost. A new hash is then generated on the next query. This behavior will prevent an attacker from learning several responses that are known to be from the same user, which is essential for the randomization technique proposed in this paper to be effective.

Security researchers are rightfully wary of assumptions that limit attacker capabilities, and we must be careful to not overestimate the security of a solution in situations where these limits on attackers may not hold. On the other hand, when such assumptions can be established they enable new types of solutions. The approach of assuming restrictions on attacker capabilities to enable otherwise unattainable security properties was previously used by Bailey, et al. [3]. In their case, the goal was to find a protocol that can provide both public guarantees that there are no covert channels and identification privacy. The two goals are incompatible, and it is provably impossible to attain both with a strong attacker model. By restricting the attacker model to an attacker who can only sporadically interact with a tag, they were able to develop a protocol that provides both identification privacy and public guarantees that there are no covert channels. In our case, limiting the number of interactions an attacker can have with a known tag enables a protocol with enhanced privacy at limited cost. For many scenarios, this threat model is realistic given the capacitor hardware modification; in Section 3.3, we analyze the impact of relaxing this threat model.

B. Definition

The goal of a *private identification protocol* is to enable legitimate readers to correctly identify users, while preventing rogue readers and eavesdroppers from learning user identities. Private identification protocols should provide low levels of information disclosure and high levels of indistinguishability among users.

A private identification protocol takes two inputs: a secret key, s, and a random nonce, r, that is produced on the identification token and send to the reader. It produces an output, h, that can be used by a legitimate reader to identify the user. The output is typically a one-way hash that hides the key: $h \leftarrow P(s,r)$.

The legitimate reader has a set of secrets, S, and must be able to efficiently determine s given h and r. Rogue readers do not have access to S. Without knowledge of S, obtaining h and r must leak very little information about s. The legitimate reader secret, S, could be a set of keys (as in the hash protocols discussed in this paper), or a single private key (as in a public-key protocol).

The first requirement for private identification protocols is correctness. A legitimate reader should be able to efficiently and correctly identify the sender, s, given knowledge of S, h, and r. For scalability, the search for the correct key should be fast on average and should grow much slower than linearly in the number of users.

The second requirement is privacy, which we formalize using the distinguishability game. A private identification protocol must allow an attacker at most a small advantage in the following game:

Distinguishability Game. The attacker is given four values: two input nonces, r_0 and r_1, and two responses, x_0 and x_1. The attacker is then asked to decide between two different cases for how these values were generated:

1) One secret key, s: $h_0 = P(s,r_0)$, $h_1 = P(s,r_1)$.
2) Two secret keys, $s_0 \neq s_1$: $h_0 = P(s_0,r_0)$, $h_1 = P(s_1,r_1)$.

In the first case, the protocol is run twice with the same key but different nonces. In the second case, two keys and nonces are randomly chosen and the protocol is run on different keys and nonces. In each case, the distinguisher only gets to see the random nonces and the protocol output but not the secret keys. The distinguisher then has to decide whether two given responses were generated from two different secrets (in which case it outputs **0**) or using one secret (in which case it outputs **1**).

For all polynomial time distinguishers, $D(\cdot)$, there exists a bound, ε, on the probability that two users with different keys can be distinguished. The advantage, ε, with which the best distinguisher can tell the two cases apart corresponds to the maximum probability with which an attacker can distinguish two different users on average:

$$\left| \begin{array}{l} Pr[k \leftarrow U_n;\ r_1,r_2 \leftarrow U_n;\ h_1 \leftarrow P(k,r_1); \\ \quad h_2 \leftarrow P(k,r_2) : D(r_1,h_1,r_2,h_2) = 1] \\ -Pr[k_1,k_2 \leftarrow U_n;\ r_1,r_2 \leftarrow U_n;\ h_1 \leftarrow P(k_1,r_1); \\ \quad h_2 \leftarrow P(k_2,r_2) : D(r_1,h_1,r_2,h_2) = 1] \end{array} \right| \leq \varepsilon$$

(U_n is a value randomly chosen from the uniform distribution over all strings of length n).

For a system to be considered private, we need ε to be small. We define a protocol to provide *polynomial privacy* if and only if the attacker advantage decreases faster than any polynomial in the key length, n. This is given when for every polynomial function $p(n)$ there exists an m for which for all $n > m : \varepsilon \leq p(n)^{-1}$.

For a protocol to also provide authentication, an attacker must not be able to spoof a user's response without knowing the secret keys, even after seeing many of the user's responses. Authentication follows trivially from private identification in the tree protocol, which is introduced in Section II-D, by adding a nonce selected by the server. In this paper, we concentrate on improving private identification. The results apply equally to private authentication, because our approach only alters the higher levels of the tree, while authentication only happens at the lowest level of the tree.

C. Measuring Privacy

The advantage ε represents the privacy of a system, but there is no fixed threshold value of ε that separates private from non-private. Instead the attacker advantage necessary to put a system at risk will vary across systems as well as attackers. Instead of arbitrarily selecting an ε bound, we provide a way of measuring privacy through *information leakage* [15]. Based on this measure, the designer of a system will have to decide what level of privacy is required (and affordable). The metric enables protocol designers to trade-off between scalability and privacy.

Information leakage measures how accurately an attacker can distinguish groups of users. It is calculated as the loss of entropy over perfect privacy (defined as: $\varepsilon = 0$) averaged over all users of a system. The loss in entropy is computed as the entropy of the sizes of groups of users that are distinguishable:

$$I = \sum_i p_i \cdot \log_2 \left(p_i^{-1}\right)$$

where p_i is the fraction of users in the ith group. If, for example, an attacker can distinguish 3 groups of 25, 25, and 50 users, the average information leakage is $L = 2 \cdot \frac{1}{4} \cdot \log_2(4) + \frac{1}{2} \cdot \log_2(2) = 1.5$ bits.

The attacker advantage, ε, can be expressed in terms of information leakage. If on average x bits of information are learned from users of a system, the attacker advantage of distinguishing two users is the probability that these users do not share the same x bit identifier; that is $\varepsilon \geq 1 - 2^{-x}$.

If N users are evenly distributed over 2^x groups, each user is in a group of size $2^{-x}N$ and $\varepsilon = 1 - 2^{-x}$. If the groups are distributed differently, ε is larger. If, for example, there exist groups of two sizes, $(2^{-x} + \alpha) \cdot N$ and $(2^{-x} - \alpha) \cdot N$, then $\varepsilon = 2^{-x} + 2 \cdot \alpha^2$. As long as the deviation from the uniform distribution, α, is small, the attacker advantage is very close to the bound. Given this conversion between information leakage and attacker advantage, we can estimate the attacker advantage, ε, caused by an information source by measuring its information leakage. This paper uses information leakage as the privacy measure since it converts to distinguishability as defined in the distinguishability game but is usually more intuitive to calculate.

D. Protocols

Private identification protocols have been proposed that provably provide polynomial privacy, but cause extensive computational overhead on the back-end server since they require the back-end to try every possible key [22]. Alternatives that are more scalable sacrifice either availability or strong privacy. Those protocols with limited availability maintain and synchronously update some shared state on the tag and server [16]. Too many unauthorized read attempts bring this state out of sync and the tag is effectively lost from the database. The other possible trade-off for more scalability sacrifices some privacy by sharing secrets among different users. While many of the ideas in this paper could be applied to other protocols, we focus on extending Molnar and Wagner's tree-based hash protocol in which keys are organized in a tree of secrets [12].

The tree-based hash protocol extends the basic hash protocol. In the basic hash protocol, each user is assigned a single unique key [22]. When queried, the user responds with a random nonce and the keyed hash of that random number: $\langle H(s,r), r \rangle$ where $H(\cdot,\cdot)$ is a one-way function, s is a secret key, and r is a random nonce. To identify the user, the server hashes the nonce under all keys in the database until it finds a match. Assuming a strong one-way function, the basic hash protocol provides polynomial privacy (e.g., $\varepsilon < 1/p(n)$ over all polynomial functions, $p(n)$) but does not scale well, which results in computationally prohibitive overhead for large systems.

In the more scalable tree protocol, several secrets are assigned to each user [12]. The secrets are structured in a tree with the users as the tree leaves. A user 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 users). When queried, user t_i responds with:

$$\langle H(s_{i,1}, r_1), r_1, H(s_{i,2}, r_2), r_2, \cdots, H(s_{i,d}, r_d), r_d \rangle$$

The server finds the matching secret on each level in the same way it did it in basic hash protocol. By identifying the correct branch on each tree level, a leaf is reached which uniquely identifies the user. This tree-based hash protocol scales well beyond billions of users. The drawback of the protocol, however, is that secrets are shared among several users and extracting the secrets from some users potentially allows tracking others. An attacker can uniquely identify a user with higher probability when more secrets of that user are known. In the standard tree protocol, a tree with a constant branching factor at each level is used. Previous work has shown that varying the branching factor for the different levels improves privacy [11][1][15]. We assume an already optimized tree protocol (typically with only two levels) and further improve privacy through randomization.

III. Randomized Tree Protocol

The privacy of probabilistic privacy protocols derives from the fact that an attacker only ever knows a small fraction of the keys in a system while the legitimate reader knows all keys. This gap in the ability to distinguish users can be amplified by adding noise to user responses. The noise blurs the borders between groups of users that the attacker would otherwise be able to distinguish.

Our technique for improving the tree protocol's privacy is simple: some bits of the user-generated hashes are randomly flipped before being sent to the server. To enable legitimate readers to still uniquely identify each user with total correctness, the last level of the tree is not randomized. Whereas before an attacker in possession of the relevant shared keys could deterministically identify a user as being a member of a single group, the randomization means an attacker can only determine the probability that the user is in each group.

The randomization never leads to false identifications, because the last round that uses the unique secret of the tag is not randomized. The legitimate reader will always be able to correctly identify a tag, but some wrong tree branches might be evaluated before the correct branch thereby increasing the identification cost. Randomization enables privacy levels anywhere between the deterministic tree-protocol and the basic hash protocol at varying costs.

In a simple instantiation of our randomization technique, the user generates a nonce, then hashes that nonce and a secret key, and finally flips every bit of the result with some probability p. Any response could hence have been generated by any user with some probability. If p is chosen as $\frac{1}{2}$, even the legitimate reader has no advantage in identifying the user over trying all keys on the lowest tree level until the right one is found. If p is chosen suitably, it can be close enough to $\frac{1}{2}$ to provide little information to an attacker operating a rogue reader, but far enough from $\frac{1}{2}$ to provide useful information to a legitimate reader. The legitimate reader only has to try a few groups to find the right user, while an attacker is left with large uncertainty and cannot determine to which of the groups a user belongs.

Section III-A analyzes the privacy properties of the proposed scheme, and Section IV-A estimates the costs a legitimate reader incurs for different choices of p. In Section III-B, we present and analyze a variation on the simple randomization technique that provides improved privacy-cost tradeoffs by using selective randomization. Our protocol is designed around the strong assumption that an attacker cannot obtain multiple reads known to be from the same tag. In Section III-C, we consider how privacy is reduced when an attacker can perform multiple reads.

A. Privacy Analysis

We analyze a two-level tree that has unique secrets on the second level; hence, no information is leaked from this level. We assume the strongest possible attacker who knows all secrets on the first level but none of the second level secrets (the secrets on the second level are not shared among tags and hence not stealable). A realistic attacker is not likely to get access to all the first-level secrets, as that would involve physically compromising tens of thousands of tags.

Figure 1 depicts the probability distribution that reflects where in the tree a given user resides as seen by an attacker. Since the attacker knows all secrets on the first level of the tree, each user can be placed into one of the tree branches when using the deterministic tree protocol. This decreases the number of possible tree positions from N to $\frac{N}{k}$ and leaks $\log_2 k$ bits of information, where k is the spreading factor on the first tree level. When randomization is used, the attacker only learns with what probability the user resides in the different branches.

The amount of entropy (and information leakage) depends on the probability that an attacker guesses that the correct secret was used (marked as a_1 in Figure 1) and the probability that the attacker guesses that any of the other secrets was used (a_2 in the graph). These *correct guess* and *wrong guess probabilities* are directly related to the degree of randomization, p. The correct guess probability is the chance that a received response corresponds to the hash output using an assumed key plus randomization. The wrong guess probability corresponds to the case that the response corresponds to the output from some other key. The closer these two probabilities are, the more privacy is provided since an attacker can no longer decide whether or not a certain key is used.

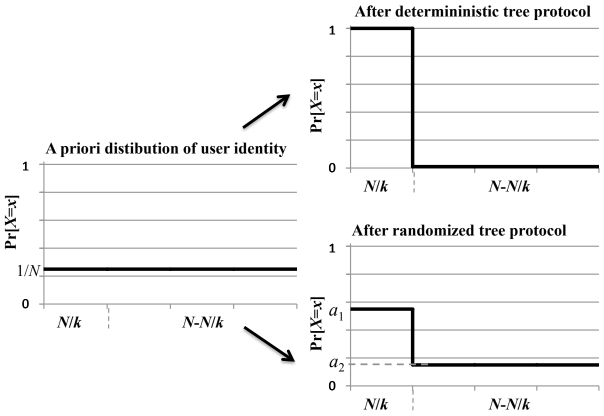

Figure 1. Change in probability distribution of user identity as seen by attacker compared for deterministic and randomized tree protocol. Tree with N users, spreading factor k.

The correct guess probability is calculated as the likelihood that a given level of noise is the result of randomization summed over all possible levels of noise. Each such chance corresponds to the probability that a certain level of noise was produced by a binomial distribution, $Binom\left(i,n,p\right)$, which stands for the chance that i of n bits in the output are flipped for randomization of degree p.

The wrong guess probability is conversely derived from an unbiased distribution, $Binom\left(i,n,\frac{1}{2}\right)$. The ratio between correct and wrong guess probability, $r = \frac{a_1}{a_2}$, is:

$$r = \sum_{i=0}^{n} \frac{Binom\left(i,n,p\right)^2}{Binom\left(i,n,\frac{1}{2}\right)}$$

Average Entropy. The average entropy of tags in the randomized tree protocol as seen by an attacker is (see the Appendix for a proof of this result):

$$E = \log\left(N\right) - \log\left(k\right) + \log\left(k+r-1\right) - \frac{r}{k+r-1} \cdot \log\left(r\right)$$

The first term is the maximum entropy of a group of size N; the second term is the amount of entropy given up by the deterministic tree protocol when the attacker knows all secrets on the first level; and the remaining two terms describe the entropy gain through randomization. The border cases correspond to the linear protocol ($r = 1$, no information leaked) and the tree protocol ($r = \infty$, completely deterministic grouping of users).

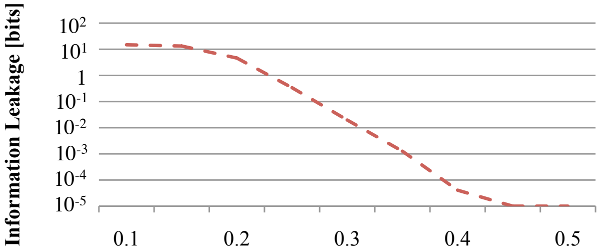

Figure 2. Information leakage decreases as randomization increases.

Figure 2 shows the different amounts of information leakage achieved by varying the degree of randomization. The amount of disclosed information decreases roughly exponentially with the degree of randomization (and the attacker advantage, ε, decreases roughly linearly). In the range $0.2 \leq p \leq 0.4$, the amount of information leakage, I, drops exponentially as $I = 4.7\left(1.78 \times 10^{-24}\right)^{p-0.2}$. This exponential drop-off is a conservative lower bound since we are assuming that the attacker knows all secrets on the first tree level. For high levels of randomization, the information leakage drops to virtually zero. For a two-level tree with one billion tags, for example, the information leakage drops to 1/10,000th of its original value for $r = 19$.

B. Selective Randomization

To reach more points in the design space, most of which are superior to what the simple randomization can achieve, we introduce an extension to the randomization scheme. In this *selective randomization* scheme, we first select a fixed size subset of bits and then flip each bit in this set with a certain probability. On each read, the tag randomly selects a new set of $p_1 n$ of the n bits for randomization and then flips each of these bits with probability p_2. The simple randomization analyzed previously corresponds to the case where $p_1 = 1$.

Selective randomization leads to a distribution with the same expected number of flipped bits as the simple randomization with $p = p_1 p_2$, but the actual distribution is more concentrated around this average. In particular, no value with more than $p_1 n$ flipped bits can be reached. This constraint could help the attacker in that some users are known to not have generated some responses. For well-chosen p_1 and p_2, however, the probability that at least a few of the wrong secrets could have generated any given response is very high.

We still assume the worst-case scenario where the attacker knows all secrets on a given tree level. The best attacker strategy for identifying the correct secret is to compute the probability for each of the secrets that a certain deviation was caused by added noise and choose the secret with the highest probability. For a single read that deviates from the computed response in x bits, the probability that the deviation was caused by added noise is described by the binomial distribution: $Pr_{right}(x) = Binom(x, n \cdot p_1, p_2)$.

Conversely, the probability that a response generated under a different secret randomly matches in x bits is: $Pr_{wrong}(x) = Binom(x, n, 0.5)$.

The ratio between the correct secret's probability of having a certain deviation and the probability that any of the secrets have that deviation is:

$$\frac{Pr_{right}(x)}{Pr_{right}(x) + (k-1) \cdot Pr_{wrong}(x)}$$

The average probability that the correct secret is chosen after a single read is this ratio multiplied by the probability that this deviation occurs, $Pr_{correct}(x)$. Summing over all possible deviations:

$$Pr_{identification} = \sum_{i=0}^{n \cdot p_1} \left(\frac{Pr_{correct}(i)^2}{Pr_{correct}(i) + (k-1) \cdot Pr_{wrong}(i)} \right)$$

The equation assumes independence of the different secrets, which is approximately true when k is large. As illustrated in Section IV-A, selective randomization achieves a cost-privacy trade-off superior to that of simple randomization.

C. Multiple Readings

Our analysis has so far assumes an attacker cannot obtain multiple readings that are known to come from the same tag. If an attacker can learn several such responses, the effect of the randomization is diminished, potentially to the point where the randomized tree protocol provides no privacy advantage.

In order to calculate the probabilities for the case where the attacker combines several readings, we first have to convert the binomial distributions into more flexible normal distributions. The binomial distributions are closely approximated by normal distributions with expected values $\mu_{correct} = np_1 p_2$ and $\mu_{wrong} = n \cdot \frac{1}{2}$; and standard deviations $\sigma^2_{correct} = n \cdot p_2 \cdot (1 - p_2)$ and $\sigma^2_{wrong} = n \cdot \frac{1}{4}$.

The effect of multiple reads can be expressed as the average sum of several such normal distributions (one for each read). According to the *weak law of large numbers*, the average sum of r equally distributed normal distributions is also a normal distribution with the same expected value, but r times smaller standard deviation.

The probability that the correct secret matches in x bits averaged over r reads is approximately:

$$Pr_{right}(x, r) = Normal(x, \mu = np_1 p_2, \sigma^2 = \frac{np_2}{r}(1 - p_2))$$

Assuming an incorrect secret, the probability that the responses randomly match on average in x bits is:

$$Pr_{wrong}(x,r) = Normal\left(x, \mu = n \cdot \frac{1}{2}, \sigma^2 = \frac{1}{r} n \cdot \frac{1}{4}\right)$$

Substituting these probabilities in the average entropy equation (from Section III-A) provides the average probability that an attacker can identify the correct secret (and hence the correct group) as a function of the number of reads.

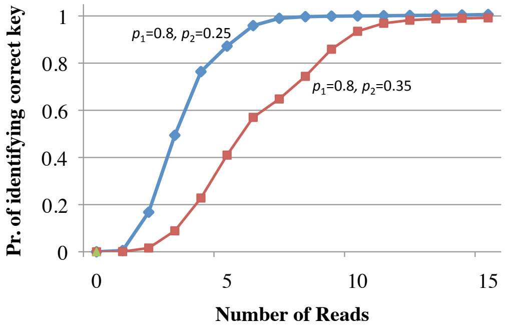

Figure 3. Effect of multiple readings on the level of privacy provided by the selective randomization tree protocol relative to the identification success when only a single reading is considered.

The probability of successfully identifying the correct group when combining multiple reads is depicted in Figure 3 for the two example parameterizations of the selective randomization. For both parameterizations, the effect of multiple reads quickly amortizes the privacy benefits of the randomization; over half of the benefits are lost after 4-6 reads. After a larger number of reads, the effect of randomization is completely lost. Note, that these are worst-case estimates, where the attacker knows all secrets on one tree level and only a single RFID tag is present in the reader field. If the attacker does not know some of the secrets or multiple tags are present at the same time, the negative effect of multiple reads is much less dramatic. As argued in Section II-A, limiting the number of unique reads in each location can be achieved for RFIDs.

IV. Cost Analysis

The computational cost of each legitimate identification grows with the level of randomization. Hence, our protocols provide an easy way to trade-off privacy and cost: increasing the degree of randomization increases privacy at the cost of increased server workload. In Section IV-A, we simulate protocol runs to estimate the cost of various parameterizations of the proposed protocols. In Section IV-B, we provide a closed-form approximation of the cost of randomization.

A. Simulated Experiments

Adding noise increases the reader cost of each identification. This verification overhead grows as more bits of the user's responses are randomized, as shown in Figure 4. In the unmodified tree protocol with a tree of height d and spreading factor k, an average of $\frac{1}{2}kd$ hashing operations are needed for every identification. When adding randomization, in the very unlikely worst case, the entire tree is evaluated for a single identification (a sensible implementation would cut off the search once the probability drops below some threshold to avoid searching the entire tree on a misread).

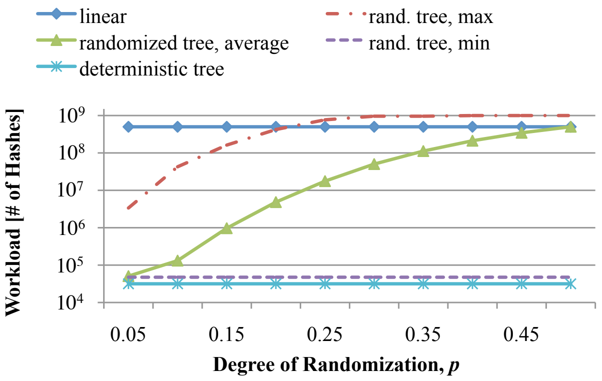

Figure 4. Workload required by different protocols for each identification. System with 1 billion users, tree with depth 2. Values are averaged over 100k simulations each.

To estimate the expected cost of the randomization, we simulated the server workload for a large number of possible parameters. The server follows a simple depth-first search strategy in which the branches are evaluated in order of their initial probability of containing the match. We choose this search strategy merely for its simplicity, while more adaptive strategies may lead to lower costs. For the two-level tree, which is optimal for many applications [1][15], there is no difference between the simple and adaptive strategies.

The average cost grows roughly exponentially with the degree of randomization as depicted in Figure 4 and information leakage decreases exponentially with randomization. Therefore, the trade-off between information leakage and cost is roughly linear.

The randomized tree protocol provides design options spanning the whole range of privacy and scalability options between the linear protocol and the deterministic tree protocol. The design space is described by the tradeoff between randomization and information leakage as depicted in Figure 2 and the tradeoff between randomization and cost in Figure 4. The resulting tradeoff curve is shown in Figure 5. All figures are for a system with one billion users and a tree with two levels for which the attacker has acquired all first-level keys.

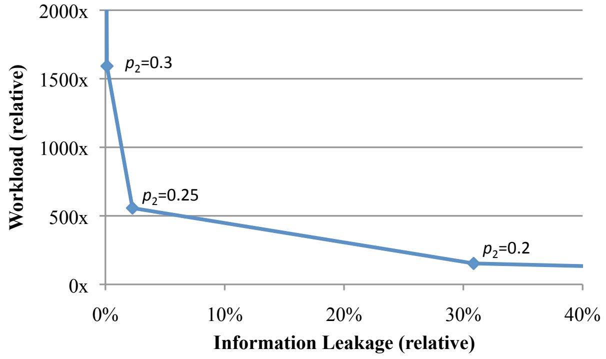

Figure 5. Design space for *simple randomization* ($p_1 = 1$). System with one billion users, two-level tree. Values are averaged over 100k simulations each.

Selective randomization provides more points in the design space, many of which are more useful than those provided by simple randomization. We calculated the amounts of entropy that the selective randomization preserves and simulated the expected cost for many choices of p_1 and p_2. The resulting options of additional cost versus decreased information leakage is depicted in Figure 6. The design space includes one design point, $p_1 = 0.8, p_2 = 0.25$, where only $\frac{1}{50}^{th}$ as much information is leaked when compared to the deterministic tree (i.e., information leakage is 98% lower) and cost increases 22 times; another design point, $p_1 = 0.8, p_2 = 0.35$, decreases information leakage to $\frac{1}{2500}^{th}$ of its original value and increases cost by a factor of 304.

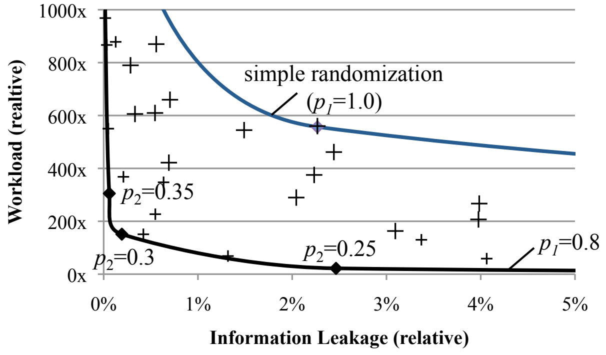

Figure 6. Design space for *selective randomization*. Each cross corresponds to one p_1, p_2 choice. The lower line corresponds to $p_1 = 0.8$ and various choices for p_2, while the upper line corresponds to the simple randomization ($p_1 = 1$). System with one billion users, two-level tree. Values are averaged over 100k simulations each.

B. Closed-Form Analysis

To find the overall cost of an identification, we calculate the number of leaf groups that need to be evaluated until the correct leaf group is found. Note, that we are not considering the small cost of evaluating the higher tree levels (typically, just one level) to further simplify the analysis.

For a received response that deviates in x bits from the n-bit hash of the correct secret, the probability that a wrong secret matches in more bits is calculated as the cumulative distribution function (CDF) of random matching, $(1 - Binom_{CDF}(x, n, 1/2))$. The number of secrets evaluated in each group before the right secret is evaluated for a given deviation, is therefore this probability multiplied with the number of wrong secrets in the group, $(k-1)$.

The average number of evaluated secrets is this number multiplied by the chance that the assumed deviation occurs, $Binom(x, n \cdot p_1, p_2)$, summed over all possible deviations. Finally, the total number of groups that need to be evaluated is calculated as the number of groups that need to be evaluated on each level exponentiated by the number of levels:

$$\text{Cost} = [\textstyle\sum_{i=0}^{n \cdot p_1} Binom_{PDF}(i, n \cdot p_1, p_2) \cdot (k-1) \cdot (1 - Binom_{CDF}(i, n, 0.5))]^{(d-1)} \cdot k_{last}$$

This gives the expected number of hashes that must be performed by a legitimate reader to identify a tag using the selective randomization protocol.

C. Feasibility

We consider a system with one billion tags (e.g., RFID tags in credit cards), each of which knows two secrets: one from the first level of a tree that is shared with some of the other tags and a unique secret on the second level. For simplicity, we assume that the spreading factor is the same on both levels at about 32,000. The attacker is conservatively assumed to know all the secrets on the first level. In the deterministic key protocol, the reader computes 32,000 hashes on average for each read. Randomization with $p_1 = 0.8, p_2 = 0.35$ increases this workload by a factor by 304 to 10 million hashing operations and lowers the information leakage by 99.96%.

State of the art implementations of hash functions such as SHA-1 and MD6 provide several Gbit/s [21] [6] of throughput, which corresponds to tens of millions of hashing operations per second. A hashing computer built from many of these chips can execute several billion hashing operations per second. For the system with a billion tags, this would support authenticating several 100,000 tags per second on a single server. At these speeds the bottleneck of the authentication process moves from cryptography to database access. Smaller cryptographic functions such as EnRUPT may well provide the cryptographic strength needed for probabilistic privacy and can save another order of magnitude in server cost [6]. Another alternative is to use probabilistic hash functions such as HB protocols [10] that can potentially build a very low-cost one-way function.

Public key cryptography that could also be used to provide privacy compares unfavorably to all symmetric-key alternatives (besides exceeding the implementation cost of RFID tags). In comparison to symmetric one-way functions, public key cryptography such as Elliptic Curve Cryptography or RSA is much more expensive in hardware. FPGA implementations of RSA are six to seven orders of magnitude less efficient per area and time than symmetric-key alternatives [20]. On the other hand, only a single RSA operation is required for each identification.

A novel low-cost public key cipher specifically designed for RFID is based on the Rabin scheme [17]. The scheme can be implemented on RFID tags much smaller than alternatives like RSA while the server cost is comparable to that of RSA. The implementation size on an RFID tag is still much larger than that of a one-way function as needed in our protocol. In particular, the scheme requires a one-way function and a random number generator as building blocks. The scheme enables an elegant public key management, but has higher implementation cost and server cost.

V. Related Work

The challenge of scalable cryptography has previously been addressed in several contexts such as preventing piracy in multicast networks such as Pay-TV. Multicast security has a different threat model but is conceptually close to the question we consider. For one protocol based on a tree of secrets that allows for counterfeit Pay-TV cards to be linked to the subscriber that leaked access credentials, Poovendran and Baras derive an optimal setup using an entropy-based metric similar to the one we use [18].

Randomizing user responses was previously used to achieve privacy in RFID systems by the HB family of protocols that were originally developed by Hopper and Blum to support authentication by humans without computer assistance [8]. These protocols use only very basic mathematical operations to create a hash function and achieve one-wayness by randomly flipping some of the response bits. The security of the HB hash functions relies on the hardness of the *learning parity with noise* (LPN) problem that has not conclusively been shown to be hard. A first attempt to make the HB protocols secure against active attackers was proved secure in a limited attacker model [10], but later shown to be vulnerable against very practical attacks that are outside of the scope of the proofs [4].

None of the existing attacks, however, apply to the use of hash function in our protocol and to private identification protocols in general where the hash input is randomly chosen by the user and therefore cannot be influenced by an attacker. Improved variants of the function have been proposed that also defeat these attacks [13][5]. Since HB hash functions only require very basic arithmetic operations, the implementation overhead on an RFID tag is virtually zero. All of the HB protocol variants, however, require a significant number of rounds for each hashing operation and hence have a high communication overhead. This overhead may be acceptable in applications such as building access control where identification can take up to a second, but is not acceptable for item-level product tags. Using an HB hash function in our protocol leads to a very low-cost identification protocol for RFIDs.

VI. Conclusions

The proliferation of tiny devices incorporating unique identities with limited computing capabilities motivates the need for cheap private identification protocols. We present one such protocol that can be implemented cheaply on small devices. The randomization of user responses in our protocol provides an effective design trade-off that lowers the amount of information leakage in exchange for a reasonable increase in server workload. Levels of information leakage close to zero can be achieved at modest server cost (i.e., 99.8% privacy increase at 150x cost increase), while staying much below the cost of alternatives such as public-key protocols.

References

[1] Gildas Avoine and Tamás Holczer István Vajda Levente Buttyán. Group-Based Private Authentication. In *International Workshop on Trust, Security, and Privacy for Ubiquitous Computing*, 2007.

[2] Gildas Avoine and Philippe Oechslin. RFID Traceability: A Multilayer Problem. In *Financial Cryptography*, 2005.

[3] Daniel Bailey, Dan Boneh, Eu-Jin Goh, and Ari Juels. Covert Channels in Privacy-Preserving Identification Systems. In *ACM Computer and Communications Security Conference (CCS)*, 2007.

[4] Henri Gilbert, Matthew Robshaw, and Hervé Sibert. An Active Attack Against HB+ - A provably Secure Lightweight Authentication Protocol. In *IEE Electronic Letters*, 2005.

[5] Henri Gilbert, Matthew J.B. Robshaw, and Yannick Seurin. HB#: Increasing the Security and Efficiency of HB+. In *EuroCrypt*, 2008.

[6] Luca Henzen, Flavio Carbognani, JPA, Sean O'Neil, and Wolfgang Fichtner. Vlsi implementations of the cryptographic hash functions md6 and irrupt. In *IEEE ISCAS*, 2009.

[7] Thomas Heydt-Benjamin, Daniel Bailey, Kevin Fu, Ari Juels, and Tom O'Hare. Vulnerabilities in First-Generation RFID-enabled Credit Cards. In *International Conference on Financial Cryptography and Data Security*, 2007.

[8] Nicholas J. Hopper and Manuel Blum. A Secure Human-Computer Authentication Scheme. In *ASIACRYPT*, 2001.

[9] Xu Huang. Quantifying Information Leakage in RFID Systems. In *10th International Conference on Advanced Communication Technology*, 2008.

[10] Ari Juels and Stephen Weis. Authenticating Pervasive Devices with Human Protocols. In *Advances in Cryptology (CRYPTO)*, 2005.

[11] Tamás Holczer István Vajda Levente Buttyán. Optimal Key-Trees for Tree-Based Private Authentication. In *Workshop on Privacy Enhancing Technologies (PET)*, 2006.

[12] David Molnar and David Wagner. Privacy and Security in Library RFID: Issues, Practices, and Architectures. In *ACM Computer and Communications Security Conference (CCS)*, 2004.

[13] J. Munilla and A. Peinado. HB-MP: A Further Step in the HB-family of Lightweight Authentication Protocols. In *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 2007.

[14] Karsten Nohl and David Evans. Quantifying Information Leakage in Tree-Based Hash Protocols. In *International Conference on Information and Communications Security (ICICS)*, 2006.

[15] Karsten Nohl and David Evans. Hiding in Groups: On the Expressiveness of Privacy Distributions. In *International Information Security Conference (SEC)*, 2008.

[16] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic Approach to "Privacy-Friendly" Tags. In *RFID Privacy Workshop*, 2003.

[17] Yossef Oren and Martin Feldhofer. A Low-Resource Public-Key Identification Scheme for RFID Tags and Sensor Nodes. In *Second ACM Conference on Wireless Network Security, WiSec*, 2009.

[18] Radha Poovendran and John S. Baras. An Information-Theoretic Approach for Design and Analysis of Rooted-Tree-Based Multicast Key Management Schemes. In *IEEE Transactions on Information Theory*, 2001.

[19] T. Scott Saponas, Jonathan Lester, Carl Hartung, and Tadayoshi Kohno. Devices That Tell On You: The Nike+iPod Sport Kit. Technical Report 2006-12-06, University of Washington, 2006.

[20] Helion Technology. RSA and Modular Exponentiation Cores. *www.heliontech.com/modexp.htm*, 2009.

[21] Helion Technology. SHA-1 Hashing Cores. *http://www.heliontech.com/sha1.htm*, 2009.

[22] Stephen Weis, Sanjay Sarma, Ronald Rivest, and Daniel Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *International Conference on Security in Pervasive Computing*, 2003.

Appendix: Proof of Tag Entropy Theory

Theorem: The average entropy of tags in the randomized tree protocol as seen by an attacker is:

$$E = \log(N) - \log(k) + \log(k + r - 1) - \frac{r}{k + r - 1} \cdot \log(r)$$

Proof: Information leakage is defined as the average amount of lost entropy in the distribution of probabilities with which different users could have generated a given response. In the linear hash protocol and in public key protocols, this entropy is $\log N$ and the information leakage is virtually zero because all users could have generated a response with probability very close to $\frac{1}{N}$, where N is the number of users in the system. For the deterministic tree protocol with two levels of secrets, the first of which is completely disclosed to an attacker, the entropy is $\log N - \log k$ and the information leakage is $\log k$, where k is the number of branches of the first tree level.

In the randomized protocol, an attacker never learns the exact branch a user resides in but rather a probability distribution over the different branches as was illustrated in Figure 1. On average, the correct branch will have a higher probability than any of the wrong branches (which all have the same probability). The amount of lost entropy (i.e., information leakage) only depends on the difference of these two probabilities and the tree parameters N and k.

The entropy of the overall distribution is the weighted sum of the entropies of the tree branch that contains the user (E_1) and of all other branches (E_2):

$$E_1 = -a_1 \cdot \log(a_1) \quad E_2 = -a_2 \cdot \log(a_2) \quad r = \frac{a_1}{a_2}$$

$$\begin{aligned}
E &= \tfrac{N}{k} \cdot E_1 + N \cdot \left(1 - \tfrac{1}{k}\right) \cdot E_2 \\
&= \tfrac{k}{k+r-1} \cdot \left(\tfrac{1}{k} \log(a_1) - \log(a_1) - \tfrac{r}{k} \log(r \cdot a_1)\right) \\
&= -\log(a_1) + \tfrac{r}{k+r-1} \cdot \log(r) \\
&= -\log\left(\tfrac{1}{N} \cdot \tfrac{k}{k-r-1}\right) + \tfrac{r}{k+r-1} \cdot \log(r) \\
&= \log(N) - \log(k) + \log(k + r - 1) - \tfrac{r}{k+r-1} \cdot \log(r)
\end{aligned}$$

■