Improved Anonymity for Key-trees

Thijs Veugen[1,2] and Michael Beye[1]*

[1] Information Security and Privacy Lab, Faculty of Electrical Engineering,
Mathematics and Computer Science, Delft University of Technology, The Netherlands
M.R.T.Beye@tudelft.nl
[2] Technical Sciences, TNO, The Netherlands
thijs.veugen@tno.nl

Abstract. Randomized hash-lock protocols for Radio Frequency IDentification (RFID) tags offer forward untraceability, but incur heavy search on the server. Key trees have been proposed as a way to reduce search times, but because partial keys in such trees are shared, key compromise affects several tags. Buttyán et al. have defined measures for the resulting loss of anonymity in the system, and approximated their measures by means of simulations. We will further improve upon their trees, and provide a proof of optimality. Finally, an efficient recursive algorithm is presented to compute the anonymity measures.

Keywords: RFID, hash-lock protocol, key-tree, anonymity, anonymity set, authentication delay

1 Introduction

We consider the problem of authenticating many Radio Frequency IDentification (RFID) tags through randomized hash-lock protocols, in an efficient way. The tags are authenticated towards the reader through a challenge-response mechanism. Each tag authenticates itself using some secret key combined with a random value (*nonce*), and to authenticate the tag, the reader will have to check the keys of all tags in order to find a match. Since this task is very intensive for the reader, an authentication tree is used. Each leaf of the tree represents a tag, and each edge corresponds to a specific key. Every tag is assigned the keys that lie on its path from the root of the tree (see Figure 1). During the authentication protocol, a tag is authenticated step by step, i.e. edge by edge, such that the computational load of the reader, and thus the total authentication time, is lowered.

However, the authentication mechanism should still remain secure. If hardware-level tampering is taken into account, keys that were assigned to compromised tags can become known to the adversary. Because partial keys are shared between neighboring tags in the tree, several additional tags may be partially

[*] Part of this research was performed at TNO for a master's thesis for the University of Utrecht (UU). Special thanks go to Gerard Tel (UU) for his advice.

broken as well. How to construct the tree such that the number of (partially) broken tags will be minimal in case of one or more compromises?

This paper considers the trade-off between efficiency (minimizing authentication time), and security (minimizing the number of partially compromised tags), of such authentication mechanisms. While Buttyán, Holczer and Vajda [4] chose to keep the number of tags equal to the number of leaves in the tree, our main contribution will be to allow it to increase.

The layout of this paper is as follows: Section 2 will outline related work, with an emphasis on Buttyán et al.'s previous work on the optimization of hash-trees. In Section 3, the optimization problem is modified resulting in an improved solution, and its effect is quantified. Finally, conclusions will be drawn in Section 4. Lengthy proofs of three theorems are found in the appendices.

2 Related Work

Hash-chain protocols are meant to provide *forward untraceability*, by updating tag IDs in a one-way manner. This way, past IDs cannot be recovered, even through tampering. Examples are OSK (by Ohkubo, Suzuki and Kinoshita in [13]) and Yeo and Kim's protocol [18]. In [2], Avoine and Oechslin suggest applying time-memory trade-offs (based on Hellmann tables [7]) to hash-chain protocols (namely OSK and an improved version thereof). Hash-chain protocols have weaknesses, including *protocol exhaustion* (when the end of a chain is reached, continued updating of tag IDs will make them traceable) and *desynchronization* (server and tag chains can become out of sync if tags are queried by third parties).

A different class of hash-based authentication schemes called *Hash-lock protocols* (due to Weis et al.) was devised to solve the aforementioned problems. Tags are locked and unlocked, using hashes of their ID as the key. The *static hash-lock scheme* [17] is vulnerable to both *replay attacks* and *tracking*, but in the same paper, Weis, Sarma, Rivest and Engels offer the *randomized hash-lock scheme* as a solution to such attacks: it adds *tag freshness* (a *nonce* generated by the tag) to prevent reader impersonation and tracking. The nonce is used as a challenge, and is hashed together with the tag's ID to form a one-time-use authentication key (the expected response). Juels and Weis [8] later added reader freshness to also prevent tag impersonation.

Note that precomputation cannot be used in these protocols, because the use of freshness makes the search space too large – one would need to compute values not only for each tag, but for each tag ID in combination with all possible nonces. Other solutions are required to reduce search complexity.

Molnar and Wagner were the first to propose using a *tree of secrets* for RFID tags [9]. Although originally used for a system built around exclusive-OR and a pseudo-random function, it can be applied to other challenge-response building blocks. Damgård and Østergaard Pedersen [5] use the same concept, but speak of *correlated keys.* Nohara, Nakamura, Baba, Inoue and Yasuura in their "K-steps protocol" ([10], also dubbed NIBY) propose to apply trees to the hash-lock setting. They use the term *group IDs* rather than correlated keys, and their trees are unconventional (being of non-uniform depth). Note that all these approaches use a sequence of group- and sub-group IDs to quickly and gradually narrow down a tag's identity. As Molnar and Wagner mention, *partial keys in such a tree should be chosen independently and uniformly from a key space of sufficient entropy.* Failure to do so would make the system vulnerable to attack. If partial keys are chosen properly, the adversary will have a large key space to search, while the owner of the system can efficiently search through a limited subspace (the actual tree).

The trade-off that exists between efficiency and security in tree-based protocols was already pointed out by Avoine and Oechslin [2], with respect to Molnar's original trees. Because tags share their partial keys, if one tag is compromised (i.e. has its memory probed through invasive tampering), an adversary learns partial keys for several other tags as well. This will enable him to decipher their responses in some of the verification steps, resulting in reduced anonymity and facilitating tracking.

A paper of particular interest is by Buttyán, Holczer and Vajda [4], where the concept of trees with *variable branching factors* is introduced, to better preserve anonymity in case of attack. Our work provides an optimization of Buttyán's solution, allowing the number of leaves in the tree to increase beyond the number of tags.

2.1 Adaptive adversaries and metric

Altough this work is dedicated to static adversaries that choose compromised tags in a random way, some interesting relations can be found with other papers on adaptive adversaries that selectively choose compromised tags possibly based on some extra (side-channel) knowledge about the tags.

As in [4], we use the average anonymity set size as a metric for the level of privacy. In this metric each (subsequent) tag is considered equally likely to be compromised and therefore suits the static adversary model. Because in the adaptive adversary model different (groups of) tags could be distinghuished, Nohl and Evans [11] propose to measure information leakage in bits (or nats) which allows quantifying the potential gain of an adversary.

In succeeding work Nohl and Evans [12] investigate the trade-off between level of privacy and the cost of protection suggesting an optimal tree of depth two. A similar tree was found in [1] by Avoine, Buttyán, Holczer and Vajda who try to further improve the balance between complexity and privacy in a new authentication protocol. In short, the tags are divided into λ groups, where each group shares a group-key. Every tag also has an ID. This group-based scheme can be seen as a tree of depth 2, where every group-ID is tried, but the last stage (unique ID) only requires one decryption instead of exhaustive search. This means that the tree can be even wider at the top than a Buttyán tree, and thus attains a higher anonymity score.

However, we choose not to follow this example because we believe that the *group-based authentication protocol* in [1] has inherent flaws. Its suspected weakness lies in the fact that the final stage of narrowing down IDs is essentially skipped (the unique ID can be simply decrypted and read). If an attacker can choose his tags with some confidence, he can very quickly remove all anonymity within the system by choosing one tag from each group. Tree-based systems still preserve some measure of anonymity in these cases.

Recently, Beye and Veugen [3] analysed the case of adaptive adversaries in trees with variable branching factors. They suggest a so called Hourglass tree that provides both efficient authentication and privacy protection against intense targeted attacks. A similar approach could be used to extend our results from static to adaptive adversaries.

2.2 Notation

In this paper we use the following notation, thereby generalizing Buttyán's notation in [4]:

- $T = \{t_1, \cdots, t_N\}$: set of all tags in the system
- N: size of T, or actual number of tags in the system
- $B = (b_1, \ldots, b_d)$: a "branching factor vector" (or tuple), representing a tree of depth d
- $\sum(B)$: shorthand for $\sum_{i=1}^{d} b_i$, or the sum over all elements in B
- $\prod(B)$: shorthand for $\prod_{i=1}^{d} b_i$, or the product over all elements in B
- N': number of leaves in the tree ($\prod(B)$), or maximum number of tags in the system, $N' \geq N$
- c: number of compromised tags
- $P(t_i)$: helper function that returns the anonymity set to which tag t_i belongs (see Definition 1)
- P_j: anonymity set j, $1 \leq j \leq \ell$
- $\bar{S}$: average size over all anonymity sets in a given configuration
- $\bar{S}_c(B)$: expected value of $\bar{S}$, averaged over all configurations containing c compromised tags in the tree with branching factor vector B (see Definition 2)
- $R(B)$: resistance to single member compromise for a tree with branching factor vector B
- $R_c(B)$: resistance to c member compromise for a tree with branching factor vector B, $R_c(B) = \bar{S}_c(B)/N'$

2.3 Buttyán Trees

Buttyán et al. [4] observed the time-anonymity trade off and noted that *narrow, deep trees allow faster search; it is wide, shallow trees that provide more anonymity.* Clearly, if many tags share the same partial keys, many tags can be excluded from the search space after each authentication stage, thus making search faster. The increased anonymity can be intuitively explained by the fact that when partial keys are shared between fewer tags, the amount of information gained by compromising a single tag is limited. Buttyán uses the concept of *anonymity sets*

(Pfitzmann and Köhntopp [14], Samarati and Sweeney [15], Díaz [6]) to quantify matters.

Definition 1. *Assume a tag t_i sends a given message m (or participates in a protocol execution). For an observer O, the anonymity set $P(t_i)$ contains all tags that O considers possible originators of m. Because all tags in $P(t_i)$ are indistinguishable to O, t_i is anonymous among the other tags in the set.*

Anonymity sets provide a sliding scale for anonymity, where belonging to a larger set implies a greater degree of anonymity. Total anonymity holds if the set encompasses all possible originators in the whole system (one is indistinguishable among all N tags in T), and belonging to a singleton set implies a complete lack of anonymity.

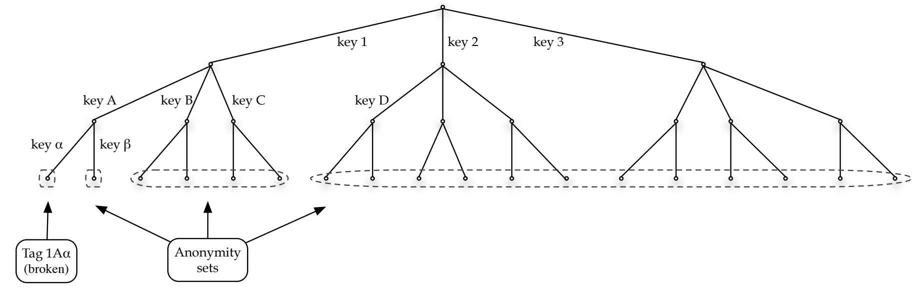


Fig. 1. Hash tree with a single broken tag [4]

To measure the level of anonymity offered by a tree, Buttyán looks at the level of anonymity provided for a randomly selected member. This *expected size of the anonymity set that a randomly selected member will belong to*, is denoted $\bar{S}$. One could also view it as *the average anonymity set size over all tags*, as shown in Equation 1 [4]. Note that $\bar{S}$ can be computed for any given scenario where a tree is broken into anonymity sets. Note that, for $c > 1$, the sizes of anonymity sets within the tree can vary, as different configurations of broken tags are formed. Configurations containing the same (number and size of) anonymity sets are considered identical, because sets can always be ordered in ascending order without loss of generality.

$$\bar{S} = \sum_{i=1}^{N} \frac{|P(t_i)|}{N} = \sum_{j=1}^{\ell} \frac{|P_j|}{N} |P_j| = \sum_{j=1}^{\ell} \frac{|P_j|^2}{N} \quad , \tag{1}$$

where $P(t_i)$ is a function that returns the anonymity set to which tag t_i belongs, P_j denotes an anonymity set and ℓ is the number of sets.

Buttyán then defines R, the *resistance to single member compromise*, as $\bar{S}$ computed for a scenario where *a single tag* is broken, and then normalizing the result (as in Samarati and Sweeney [15] generalized by Díaz [6]). Note that because we can freely order the anonymity sets, $c = 1$ leads to a single unique configuration. With its range of $[0, 1]$, $R = \frac{\bar{S}}{N}$ is independent of N, allowing for easy comparison between systems of different sizes. In the scenario of single member compromise as depicted in Figure 1, the number of anonymity sets is equal to $d + 1$.

We will refer to trees with a constant branching factor as "*Classic trees*". Buttyán proposes the use of trees with different, independent branching factors on each level, sorted in descending order as shown in Figure 1. Trees will be described by their branching factor vectors $B = (b_1, \ldots, b_d)$, where the variables b_i $(1 \leq i \leq d)$ are integers larger than 1 denoting the branching factor at level i.

Buttyán et al. in [4] reach the conclusion that the branching factors near the root contribute more to $\bar{S}$ and R. For trees with variable branching factors this means that a deep, top heavy tree can potentially outperform a shallow classic tree.

They present a *greedy* algorithm that recursively finds the branching factor vector B that maximizes R, given a number N of tags and a maximum authentication delay D_{max}. It starts with the prime factorization of N and tries to combine prime factors as long as the sum $\sum(B)$ (authentication delay) remains acceptable. An important assumption is that the number of leaves in the tree is equal to the number of tags, i.e. $\prod(B) = N$.

However, Buttyán recognizes that trees need to stand up to more than single tag compromise. We suggest to express $\bar{S}$ for the general case as follows:

Definition 2. *$\bar{S}_c(B)$ expresses $\bar{S}$ as the average over all $\binom{N}{c}$ possible distributions of c compromised members across the tag set T which consists of the $N = \prod(B)$ leaves of the tree represented by branching factor vector B.*

Our notation is a natural extension of Buttyán's $\bar{S}_{\langle - \rangle}$, directly incorporating B and c. Depending on how each successive member is picked from the tree, different anonymity sets are broken down.

3 Improved Hash-trees

Our main observation is that Buttyánś condition $\prod(B) = N$ can lead to inferior solutions. Particularly when the number N has large prime factors, resulting in a small number of candidate branching factor vectors. We prefer the condition $\prod(B) \geq N$, which we will show leads to better results. An added advantage in practice is that it allows to maintain a small buffer of extra keys (see discussion in Section 3.1). Our optimization problem now becomes:

Problem 1. Given the total number N of members and the upper bound D_{max} on the maximum authentication delay, find the vector $B = (b_1, \ldots, b_d)$ that maximizes $R(B)$ subject to the following constraints:

$$\prod(B) = \prod_{i=1}^{d} b_i \geq N, \text{ and } \sum(B) = \sum_{i=1}^{d} b_i \leq D_{max} \ . \tag{2}$$

The anonymity measure $R(B)$ used here refers to the full tree with $\prod(B) = N'$ tags, of which exactly one is compromised, i.e. $c = 1$. Theorem 3 will later show that the same holds for the anonymity measure of the partial tree with $N \leq N'$ tags.

Theorem 1. *The maximal $R(B)$ under the constraints $\prod(B) \geq N$ and $\sum(B) \leq D_{max}$ is achieved by the lexicographically largest vector B that satisfies these constraints.*

The proof of Theorem 1 is given in the Appendix. The following theorem, whose proof is in the appendix, shows how to optimize the product of a branching vector, while keeping the sum constant and ignoring the lexicographic order. The notation (3^*) is used to denote a (possibly empty) branching factor vector of arbitrary dimension consisting solely of factors 3.

Theorem 2. *Let $D \geq 2$ be a fixed integer and let $\prod_D^{max}$ be the largest product $\prod(B)$ attained by branching factor vectors B with sum $\sum(B) = D$. Then this maximal product is attained by branching factor vectors B with $\sum(B) = D$ that have one of the following shapes: (3^*), $(4, 3^*)$ or $(3^*, 2)$.*

So when searching for the vector B that optimizes $\prod(B)$, it is sufficient to search within the limited set of vectors that have one of the above described shapes. In fact, the value D mod 3 directly determines which of the three shapes should be chosen (see Appendix B).

When considering Problem 1, we know that when $D = D_{max}$ and $\prod_D^{max} < N$, there can be no solution that satisfies both constraints. On the other hand, when $\prod_D^{max} \geq N$, there is at least one solution. The obvious way to find the branching factors of the lexicographically largest solution, is to take a *greedy* approach. It means that the first branching factor is optimized first, then the second, etc. The algorithm depicted in Figure 2, which is denoted further on by Algorithm 2, takes N and D_{max} as input and solves this problem recursively [16]. A specific branching factor is allowed, when a suitable tail (according to Theorem 2) with a large enough product exists.

3.1 Consequences of Larger Trees

Algorithm 2 can lead to trees that exceed the strictly required number of leaves (with $N' > N$). We argue that this has practical advantages, but should also be taken into account when judging the anonymity of such trees.

A larger tree will allow for addition of tags at a later time, which may be desirable in practice. Ideally, creating and balancing a tree should be done only once, and therefore the tree should accommodate all the tags *ever expected to*

```
function B = VB_f(N, d_{MAX}) % VB = Veugen-Beye

Precondition: d_{MAX} > 1
Postcondition:
  B is the lexicographically largest vector satisfying
  prod(B) >= N and sum(B) <= d_{MAX}


B := [d_{MAX}]; % Start with a tree of depth one

while (prod(B) < N) and (b_1 > 2)
    b_1 := b_1 - 1; % next candidate for b1
    prod(B) := b_1 * prod^{MAX}_{d_{MAX} - b_1};
    % maximal product given first factor b_1
end;

if
    prod(B) < N         ->  "No solution exists.";
    d_{MAX} - b_1 <= 1  ->  B := b_1; % no tail left;
    else                ->  B := [b_1 VB_f(N/b_1, d_{MAX} - b_1)];
                            % find next branching factor
end;
```

Fig. 2. Recursive function for finding an optimal solution B of Problem 1

enter the system. In systems where growth is anticipated, having a larger tree that is ready for the future is good practice.

Also, since we are defending against tampering attacks, replacement of compromised tags should be taken into consideration. Replacement tags should contain *new key material*, lest they be reintroduced with keys that are already fully disclosed (immediately limiting their anonymity). Having unused leaves in the tree seems ideal for this purpose.

When choosing *which* leaves to actually use as tags (initially and for replacements), we suggest to select a sufficient number of branches at the level $d - 1$ *at random*, and to randomly initialize tags from these branches. This to create a subtree of initialized tags that is as close to the original (optimal) shape as possible, without introducing order in the system which might be exploited.

Finally note that tags corresponding to uninitialized leaves in the tree cannot be encountered by adversaries in the field. For this reason, they do not contribute to the size of the set among which targets need to be distinguished. However, given that the resistance is actually the average anonymity set size normalised per tag, it should intuitively remain roughly equal. This is formally proven in the following theorem.

Theorem 3. *If N tags are placed uniformly at random in a tree with $\prod(B) = N' > N$, then the expected resistance R_c to c member compromise satisfies*

$$R_c(B) < R_c < R_c(B) + \frac{N' - N}{N^2}$$

Because of the result of Theorem 3, whose proof is in the appendix, it makes sense to estimate the resistance R_c by the full tree resistance $R_c(B)$. This contradicts with previous work of Beye-Veugen [3] who gratuitously used a scaling factor $\frac{N}{N'}$ to adjust their anonymity measures. There is a flaw in the proof of their Theorem 4 where they pose that $E[\frac{1}{N}\sum_{i=1}^{N}\frac{|P(t_i)|}{N}] = E[\frac{1}{N}\sum_{i=1}^{N'}\frac{|P(t_i)|}{N}]$ which explains the erroneous appearance of the factor $\frac{N}{N'}$. However, with respect to their full paper it is only a minor flaw and doesn't affect their main conclusions.

3.2 Comparison of performance

Since our search space is larger than Buttyán's, our trees potentially perform better in two ways:

1. Given the same maximal delay D_{max}, we might find a lexicographically larger tree that provides better anonymity (increase in R).
2. Given the same (or at least not worse) resistance to compromise R, we might find a tree with lower $\sum(B)$ thus decreasing the authentication delay.

The results of both approaches are depicted in Table 1 where for three different categories 1000 random instances (N, D_{max}) have been generated and the results have been averaged. The intervals for the parameters N and D_{max} have been chosen such that their sizes resemble those of Buttyán.

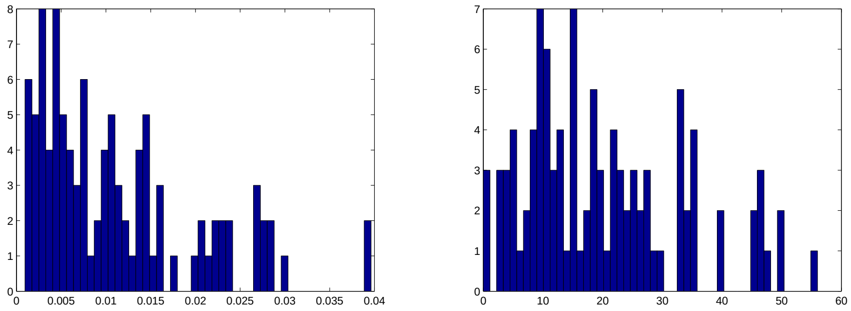
N		D_{max}		# of solvable instances		Average increase in R	Average decrease in D
min	max	min	max	Buttyán	Our work		
1000	10000	40	120	221	1000	0.0126	10.6290
10000	100000	50	150	101	1000	0.0112	20.0099
100000	1000000	60	180	46	1000	0.0160	27.1087

Table 1. Increase of performance given 1000 random instances

Remarkably, a huge number of instances turn out to be unsolvable within Buttyán's optimization problem. Analysis learns that this is due to the large prime factors of these values of N which raise the delay to an unacceptable level. Indeed, the minimally achievable delay in Buttyán's setting equals the sum of all prime factors of N. As argued in Theorem 1, our minimally achievable delay is roughly $3\log_3 N$ (when all branching factors are three, see Theorem 2) which explains that all instances are solvable within Problem 1.

To obtain better insight in our actual improvements, the performance of the 101 solvable instances with $10^4 \leq N \leq 10^5$ and $50 \leq D_{max} \leq 150$ is analyzed in more detail by two histograms showing the distribution of the increase in R (Figure 3(a)) and the decrease in D (Figure 3(b)) respectively over 50 equally sized bins. Figure 3(a) shows e.g. that we were able to increase the resistance of compromise of 2 instances by a value between 0.0392 and 0.04.

The achievable increment in R may seem modest but is comparable with Buttyán's improvement with respect to the Classic tree. The advantage will be more significant for larger values of c as shown in Figure 4(b). The attainable slump in authentication delay by our new trees can be considered substantial. So besides from the fact that many instances are unsolvable in Buttyán's setting, our trees outperform Buttyán's trees on both higher R and lower D_{max}.



(a) Resistance to compromise

(b) Delay

Fig. 3. Histograms of improved performance over 101 random instances

3.3 Multiple compromised tags

Subsection 3.2 has already shown that our proposal yields lexicographically larger B's than Buttyán's approach, and consequently better anonymity measures when $c = 1$. The computation of resistance to compromise $R_c(B)$ becomes more difficult for $c > 1$. Buttyán noted that computing $\bar{S}_c(B)$ is hard, and therefore suggested an alternative measure $\bar{S}_0$ corresponding with an even distribution of c compromised tags across T which he used to approximate $\bar{S}_c(B)$.

Proposition 1. *Although not stated explicitly in [4], $\bar{S}_0$ actually represents the worst-case choice of c compromised tags across T resulting in the minimal value of $\bar{S}$.*

Proof. Assume that we are allowed to choose tags to be compromised sequentially, with the aim to minimize the average anonymity set size. The first compromised tag leads to a unique configuration (as described further on). Each subsequent compromised tag leads to a new configuration, with more anonymity sets (of varying, decreasing size). To minimize the average set size in the *resulting* configuration, the next tag to be compromised should be chosen from (one of) the largest anonymity set(s) in the *current* configuration. When sorting anonymity sets in ascending order, we observe that this is equivalent to choosing tags (as) evenly (as possible given the tree structure) across T. By induction, our claim holds for any c. □

Buttyán[4], and Beye and Veugen [3] used simulations to approximate $\bar{S}_c(B)$, but we present an efficient algorithm for recursively computing the exact resistance and compare our approach to Classic and Buttyán trees by means of numerical computations.

Let $U_c(B) = \sum_{i=1}^{N} |P(t_i)|$ be the anonymity set size added over all tags, after c particular tags from the tree with branching factor vector B have been compromised. We would like to compute

$$\bar{S}_c(B) = \frac{\bar{U}_c(B)}{\prod(B)}$$

where $U_c(B)$ is averaged over all possible choices of c out of $\prod(B)$ tags. Note that resistance $R_c(B)$ to c-member compromise equals $\bar{S}_c(B)/\prod(B)$.

When one tag of the $N = \prod(B)$ tags of the tree with branching factor vector $B = (b_1, b_2, \ldots b_d)$ is compromised, the tree falls into $d+1$ anonymity sets (see [4] and Figure 1). The first anonymity set S_0 consists of the compromised tag, the other d sets S_j, $1 \leq j \leq d$, correspond to the subtrees with branching factor vector $(b_j - 1, b_{j+1}, \ldots, b_d)$ and therefore have size $|S_j| = (b_j - 1)b_{j+1} \ldots b_d$. This leads to the following recursive relation for computing $\bar{U}_c(B)$.

$$\bar{U}_c(b_1, b_2, \ldots b_d) = \begin{cases} \prod_{i=1}^{d} b_i^2 & \text{if } c = 0 \\ 1 + \bar{U}_{c-1}(b_d - 1) & \text{if } c > 0 \text{ and } d = 1 \\ 1 + \sum_{i=0}^{c-1} \sum_{j=1}^{d} f_i^j \cdot \bar{U}_i(b_j - 1, b_{j+1}, \ldots, b_d) & \text{if } c > 0 \text{ and } d > 1 \end{cases}$$

where the frequencies f_i^j are readily computed for $0 \leq i < c$, $1 \leq j \leq d$ by binomial coefficients:

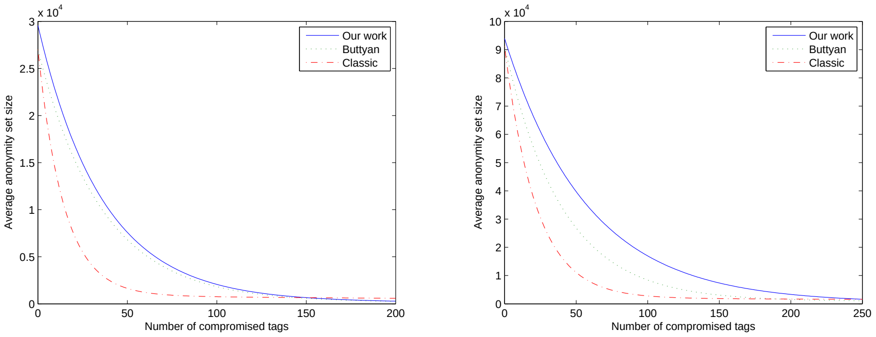
$$f_i^j = \frac{\binom{|S_j|}{i}\binom{N-1-|S_j|}{c-1-i}}{\binom{N-1}{c-1}}$$

and which represent the relative number of ways to choose i tags from anonymity set S_j and the remaining $c-1-i$ tags from the other anonymity sets. Note that $f_i^j = 0$ whenever $i > |S_j|$ or $c - 1 - i > N - 1 - |S_j|$.

We wrote a recursive MATLAB function AS_f to recursively compute $[\bar{U}_0(B), \ldots, \bar{U}_c(B)] = AS_f(B, c)$ which is available at our site [16].

While similar figures arise for larger values of N we compute, as in [4], $\bar{S}_c(B)$ for the configuration with $N = 30^3 = 27000$ and $D_{max} = 3 \cdot 30 = 90$ to make a fair comparison. The optimal tree computed by Buttyán is $(72, 5, 5, 5, 3)$, slightly improved by our Algorithm 2 to $(73, 5, 3, 3, 3, 3)$. Figure 4(a) compares their $\bar{S}_c(B)$ with the classic tree $B = (30, 30, 30)$ that has constant branching factors.

Buttyánś optimal tree for the second configuration $(N, D_{max}) = (45^3, 3 \cdot 45) = (91125, 135)$ is $(81, 25, 15, 3)$, which is further increased by Algorithm 2 to $(116, 5, 3, 3, 3, 3, 2)$. In Figure 4(b) their $\bar{S}_c(B)$ is compared with the classic tree $B = (45, 45, 45)$. We will discuss how these results relate to our hypotheses and claims.



(a) $N = 30^3$ and $D_{max} = 90$

(b) $N = 45^3$ and $D_{max} = 135$

Fig. 4. Comparison of $\bar{S}_c(B)$ for two configurations

In both figures, our tree outperforms, as expected, both the Buttyán and the classic tree in terms of $\bar{S}_c(B)$. We observe that the performance of our tree in no case drops below that of the Buttyán tree. The difference between both configurations is explained by the prime factorizations of $30 = 5 \cdot 3 \cdot 2$ and $45 = 5 \cdot 3 \cdot 3$ which gives a little more playground to Buttyán in the first configuration. In the second configuration, the gain of Buttyán's tree with respect to the Classic tree is comparable to our gain with respect to Buttyán's tree. Given that our tree has a 0.0073 higher resistance to single member compromise than Buttyán's tree, the improvement in R of the second configuration is even less than expected by a random instance as shown in Table 1. The reason is that both N's have more small prime factors than expected on average.

In each Figure we observe a turning point where the classic tree starts to outperform the other two trees. This occurs at $c \approx 2b_1$. At this point, the decrease of $\bar{S}$ slows causing the graph to *seemingly* settle into a steady minimum. We can explain this by the fact that at around this point, the last very large anonymity set is expected to have been broken down, because each top-level branch can be expected to contain at least one compromised tag. Because subsequent compromised tags then fall into smaller sets, the adversary will learn little new information; he has obtained the most important keys in the tree already. In such a worrying scenario, what little amount of anonymity tags have left depends upon the keys in lower branches. Classic trees retain slightly more anonymity, because they have larger branching factors at the bottom levels. However, given the (by then) minimal values of $\bar{S}$ overall, the *absolute* advantage is not large.

4 Conclusions and Future Work

Our proposed Algorithm 2 yields better results than Buttyán's original approach, when it comes to finding the lexicographically largest B. We have provided proof that the solution is optimal in terms of optimization problem 1. The problem that many instances are unsolvable within Buttyán's optimization setting has been solved by our modification. The solvable instances can be further optimized by our algorithm to either increase the resistance to compromise as expressed by $R_c(B)$, or to lower the authentication delay. Algorithm 2 can result in trees with $N' \geq N$, which can be advantageous in growing systems or when replacing compromised tags, and whose resistance to compromise has been proven commensurably.

For future research it might be interesting to precisely investigate to what extend $R_c(B)$ increases by lexicographically larger B for $c > 1$. Our recursive formula for computing $R_c(B)$ opens up this possibility. This could even be extended to adaptive adversary scenarios as described in [3].

References

1. Gildas Avoine, Levente Buttyán, Tamas Holczer, and Istvan Vajda. Group-based private authentication. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–6, 2007.
2. Gildas Avoine and Philippe Oechslin. A Scalable and Provably Secure Hash Based RFID Protocol. In *International Workshop on Pervasive Computing and Communication Security – PerSec 2005*, pages 110–114, Kauai Island, Hawaii, USA, March 2005. IEEE, IEEE Computer Society.
3. Michael Beye and Thijs Veugen. Privacy for key-trees with adaptive adversaries. In *7th International ICST Conference on Security and Privacy in Communication Networks (SecureComm)*, London, 2011.
4. Levente Buttyán, Tamás Holczer, and István Vajda. Optimal Key-Trees for Tree-Based Private Authentication. In *In Proceedings of the International Workshop on Privacy Enhancing Technologies (PET)*, June 2006. Springer.
5. Ivan Damgård and Michael Østergaard Pedersen. Rfid security: Tradeoffs between security and efficiency. In *Topics in Cryptology CT-RSA 2008*, volume 4964/2008 of *Lecture Notes in Computer Science*, pages 318–332, 2008.
6. Claudia Díaz. Anonymity Metrics Revisited. In Shlomi Dolev, Rafail Ostrovsky, and Andreas Pfitzmann, editors, *Anonymous Communication and its Applications*, number 05411 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.
7. M. Hellman. A cryptanalytic time-memory trade-off. In *Information Theory, IEEE Transactions on*, volume 26, pages 401–406, July 1980.
8. Ari Juels and Stephen A. Weis. Defining Strong Privacy for RFID. In *PERCOMW '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 342–347, Washington, DC, USA, 2007. IEEE Computer Society.

9. David Molnar and David Wagner. Privacy and security in library RFID: issues, practices, and architectures. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 210–219, New York, NY, USA, 2004. ACM.
10. Yasunobu Nohara, Toru Nakamura, Kensuke Baba, Sozo Inoue, and Hiroto Yasuura. Unlinkable identification for large-scale rfid systems. *Information and Media Technologies*, 1(2):1182–1190, 2006.
11. Karsten Nohl and David Evans. Quantifying information leakage in tree-based hash protocols (short paper). In Peng Ning, Sihan Qing, and Ninghui Li, editors, *ICICS*, volume 4307 of *LNCS*, pages 228–237. Springer, 2006.
12. Karsten Nohl and David Evans. Hiding in groups: On the expressiveness of privacy distributions. In *23rd International Information Security Conference (SEC 2008)*, Milan, sep 2008.
13. Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic Approach to "Privacy-Friendly" Tags. In *RFID Privacy Workshop*, MIT, MA, USA, November 2003.
14. Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 1–9. Springer-Verlag, 2001.
15. Pierangela Samarati and Latanya Sweeney. Generalizing data to provide anonymity when disclosing information. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (PODS)*, page 188, Seattle, WA, USA, 1998.
16. Thijs Veugen and Michael Beye. Matlab code for "improved anonimity of hash-trees". In *RFIDsec.* http://isplab.tudelft.nl/content/improved-anonimity-hash-trees, 2012.
17. Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In Dieter Hutter, Günter Müller, Werner Stephan, and Markus Ullmann, editors, *SPC*, volume 2802 of *LNCS*, pages 201–212. Springer, 2003.
18. Sang-Soo Yeo and Sung Kwon Kim. Scalable and Flexible Privacy Protection Scheme for RFID Systems. In Refik Molva, Gene Tsudik, and Dirk Westhoff, editors, *European Workshop on Security and Privacy in Ad hoc and Sensor Networks – ESAS'05*, volume 3813 of *LNCS*, pages 153–163, Visegrad, Hungary, July 2005. Springer-Verlag.

A Proof of Theorem 1

In this appendix we proof Theorem 1. By $B \backslash \{b_1, \cdots, b_j\}$, we denote the vector $(b_{j+1}, \ldots, b_d)$, where $1 \leq j \leq d$.

The first observation is that for an optimal B, $\sum(B) = D_{max}$, otherwise $D_{max} - \sum(B)$ could be added to any element of B without violating the constraints while increasing $R(B)$. So we assume $\sum(B) = D_{max}$ in the proof, which uses four Lemmas, similar to the Lemmas of Buttyán's work [4]. It's also clear that an optimal B will have branching factors at least 2. The first Lemma, Lemma 1, shows that a branching vector can always be improved by ordering its elements in decreasing order. Lemma 3, using some bounds from Lemma 2, shows that given two branching factor vectors, the one with the larger first element is always at least as good as the other. Lemma 4 generalizes Lemma 3 by stating that given two branching factor vectors the first j elements of which are equal, the vector with the larger $(j+1)$-st element is always at least as good as the other.

These Lemma's together show that a lexicographically larger branching factor vector will always be at least as good as the lexicographically smaller branching factor vector (in case $\sum(B) = D_{max}$), so indeed the solution with maximal $R(B)$ to Problem 1 is achieved by the lexicographically largest vector that satisfies the constraints.

Lemma 1. *Let B be a branching factor vector, and let B^* be the vector that consists of the sorted permutation of the elements of B in decreasing order. If B satisfies the constraints of Problem 1, then B^* satisfies them too, and $R(B^*) \geq R(B)$.*

Proof. Since $\prod(B)$ is not altered by the permutation, we can refer to Buttyán's proof [4] of Lemma 1. □

Lemma 2. *Let $B = (b_1, \ldots, b_d)$ be a sorted branching vector (i.e. $b_1 \geq b_2 \geq \ldots \geq b_d$). We can give the following lower and upper bounds on $R(B)$:*

$$\left(1 - \frac{1}{b_1}\right)^2 \leq R(B) \leq R(b_1) = \frac{1 + (b_1 - 1)^2}{b_1^2}$$

Proof. The lower bound is identical to Buttyán, hence the proof [4] is as well. The upper bound is an improvement w.r.t. Buttyán, and is proven as follows. Let $M = \prod(B)$, then $\prod(B \backslash b_d) = M/b_d$. We derive for $d > 1$:

$$\begin{aligned}
R(B) &= \frac{1}{M^2}\left(1 + (b_d - 1)^2 + \sum_{i=1}^{d-1}(b_i - 1)^2 \prod_{j=i+1}^{d} b_j^2\right) \\
&= \frac{1}{M^2}\left(1 + (b_d - 1)^2 + \sum_{i=1}^{d-2}(b_i - 1)^2 \prod_{j=i+1}^{d} b_j^2 + (b_{d-1} - 1)^2 b_d^2\right) \\
&= R(B \backslash b_d) - \frac{b_d^2}{M^2}\left(1 + (b_{d-1} - 1)^2\right) + \frac{1}{M^2}\left(1 + (b_d - 1)^2 + (b_{d-1} - 1)^2 b_d^2\right) \\
&= R(B \backslash b_d) + \frac{2 - 2b_d}{M^2} \\
&< R(B \backslash b_d)
\end{aligned}$$

and by recursively applying this inequality also $R(B) \leq R(b_1)$. □

Lemma 3. *Let $B = (b_1, \ldots, b_d)$ and $B' = (b'_1, \ldots, b'_{d'})$ be two sorted branching factor vectors (i.e. $b_1 \geq b_2 \geq \ldots \geq b_d$, $b'_1 \geq b'_2 \geq \ldots \geq b'_{d'}$) that satisfy the constraints of Problem 1. Then, $b_1 > b'_1$ implies $R(B) \geq R(B')$.*

Proof. We first prove the statement for $b'_1 \geq 3$. From Lemma 2 we know that

$$R(B') \leq \frac{1 + (b'_1 - 1)^2}{{b'_1}^2}$$

and

$$R(B) \geq \left(1 - \frac{1}{b_1}\right)^2 > \left(1 - \frac{1}{b'_1 + 1}\right)^2$$

which follows from the fact that $b_1 > b'_1$. A straightforward calculation shows that $(1 - \frac{1}{b'_1+1})^2 \geq \frac{1+(b'_1-1)^2}{{b'_1}^2}$ whenever $b'_1 \geq 3$, and thus $R(B) \geq R(B')$.

So the remaining case is $b'_1 = 2$. Since B' is ordered, each element of B' will equal 2. If $d' = 1$ then by our previous assumption $D_{max} = \sum(B') = 2$, but this contradicts $D_{max} = \sum(B) \geq 3$, so we know $d' \geq 2$. The resistance $R(B')$ is readily computed as $R(B') = \frac{1}{3}(2 \cdot 4^{-d} + 1)$, which will be at most $\frac{3}{8}$ (when $d' = 2$). Since $R(B) \geq (1 - \frac{1}{b_1})^2 > (1 - \frac{1}{3})^2 = \frac{4}{9}$, it follows that also in this case $R(B) \geq R(B')$. □

Lemma 4. *Let $B = (b_1, \ldots, b_d)$ and $B' = (b'_1, \ldots, b'_{d'})$ be two sorted branching factor vectors (i.e. $b_1 \geq b_2 \geq \ldots \geq b_d$, $b'_1 \geq b'_2 \geq \ldots \geq b'_{d'}$) that satisfy the constraints of Problem 1. Let j, $1 \leq j < \min(d, d')$, be such that $b_i = b'_i$ for all i, $1 \leq i \leq j$, and $b_{j+1} > b'_{j+1}$, then $R(B) \geq R(B')$.*

Proof. It is easy to show that $R(B) = \left(\frac{b_1-1}{b_1}\right)^2 + \frac{1}{b_1^2} \cdot R(B \backslash b_1)$. Therefore, since $b_1 = b'_1$, $R(B) \geq R(B')$ whenever $R(B \backslash b_1) \geq R(B' \backslash b'_1)$. By recursively applying this rule, and using Lemma 3, which shows that $R(B \backslash \{b_1, \ldots, b_j\}) \geq R(B' \backslash \{b'_1, \ldots, b'_j\})$, the proof is complete. The proof of Lemma 4 is similar to the proof of Buttyán's Lemma 4 [4]. □

B Proof of Theorem 2

This appendix contains the proof of Theorem 2.

Proof. Let B be a branching factor vector with $\sum(B) = D$. The proof is given by considering different cases.

Suppose B has a branching factor b_i equal to 1. Since $\sum(B) \geq 2$, there must be another branching factor b_j. Then, we could add b_i to b_j to increase $\prod(B)$ without modifying $\sum(B)$, meaning $\prod(B) \neq \prod_D^{max}$. Therefore, an optimal B (with $\prod_D^{max}$) contains no branching factor equal to 1.

Suppose B has a branching factor $b_i \geq 5$. Since $(b_i - 3) \cdot 3 > b_i$, we can increase $\prod(B)$ without modifying $\sum(B)$, by making an extra factor 3, meaning $\prod(B) \neq \prod_D^{max}$. Therefore, an optimal B contains only branching factors 2, 3 or 4.

Suppose B has two branching factors $b_i = b_j = 4$ $(i \neq j)$. Since $3 \cdot 3 \cdot 2 = 18 > 16 = 4 \cdot 4$, we can increase $\prod(B)$ without modifying $\sum(B)$ by changing

b_i and b_j to 3 and adding an extra 2, meaning $\prod(B) \neq \prod_D^{max}$. Therefore, the optimal B contains at most one branching factor 4.

Suppose B has two branching factors $b_i = b_j = 2$ $(i \neq j)$. Since $2 \cdot 2 = 4$, we could just as well substitute these branching factors by a single 4, making B lexicographically larger. Therefore, $\prod_D^{max}$ can be attained by at most one branching factor 2.

Suppose B has two branching factors $b_i = 2$ and $b_j = 4$. Since $2 \cdot 4 = 8 < 9 = 3 \cdot 3$, we can increase $\prod(B)$ without modifying $\sum B$ by substituting both factors by 3, meaning $\prod(B) \neq \prod_D^{max}$. Therefore, an optimal B will not contain both branching factors 2 and 4.

By considering these five cases, it follows that $\prod_D^{max}$ will be attained in one of the following cases:

1. B contains only 3's;
2. B contains one 4 and an arbitrary number of 3's;
3. B contains one 2 and an arbitrary number of 3's.

Consequently when $\sum(B) = D$, and we order the elements descendingly, $\prod_D^{max}$ will be attained by:

1. $B = (3^*)$, when $D \bmod 3 = 0$;
2. $B = (4, 3^*)$, when $D \bmod 3 = 1$;
3. $B = (3^*, 2)$, when $D \bmod 3 = 2$. □

C Proof of Theorem 3

In this appendix Theorem 3 is formally proved.

Proof. Intuitively, the average size of an anonymity set is decreased by a factor $\frac{N}{N'}$ when $N' - N$ tags are removed from the full tree. Therefore, the expected resistance should not decrease. We first proof this for $N' = N + 1$ and use the result to generalize the statement to arbitrary N.

Let $P_1, \ldots P_\ell$ be the anonymity sets of the full tree after c tags have been compromised, so $\sum_{j=1}^{\ell} |P_j| = N'$. The average anonymity set size over all tags $\bar{S}_{N'}$ will equal

$$\bar{S}_{N'} = \sum_{j=1}^{\ell} \frac{|P_j|^2}{N'}$$

Note that $\bar{S}_{N'}$ is an instantiation of $\bar{S}_c(B)$ for a particular choice of c compromised tags. When one tag is uniformly chosen to be removed, the probability that this tag is chosen from the j^{th} anonymity set will equal $\frac{|P_j|}{N'}$. So the average anonymity set size over all remaining N tags $\bar{S}_N$ will equal

$$\bar{S}_N = \sum_{j=1}^{\ell} \frac{|P_j|}{N'} \cdot \frac{1}{N} \left\{ (|P_j| - 1)^2 + \sum_{i=1, i \neq j}^{\ell} |P_i|^2 \right\}$$

We derive

$$\sum_{j=1}^{\ell} |P_j| \cdot \left\{ (|P_j| - 1)^2 + \sum_{i=1, i \neq j}^{\ell} |P_i|^2 \right\} = \sum_{j=1}^{\ell} |P_j| \cdot \left\{ 1 - 2|P_j| + \sum_{i=1}^{\ell} |P_i|^2 \right\}$$

$$= N' - 2\sum_{j=1}^{\ell} |P_j|^2 + N' \sum_{i=1}^{d} |P_i|^2$$

and thus $N \cdot N' \cdot \bar{S}_N = N' + (N' - 2)N'\bar{S}_{N'}$ or

$$N \cdot \bar{S}_N = 1 + (N' - 2)\bar{S}_{N'}$$

Finally, choose ϵ, $0 < \epsilon < N'$, such that $\bar{S}_{N'} = N' - \epsilon$, then $N' + N'(N'-2)\bar{S}_{N'} = \bar{S}_{N'} + \epsilon + N'(N'-2)\bar{S}_{N'} = \epsilon + (N'-1)^2\bar{S}_{N'}$, and thus $R_N = \frac{\bar{S}_N}{N} = \frac{1+(N'-2)\bar{S}_{N'}}{N^2} = \frac{\epsilon+(N'-1)^2\bar{S}_{N'}}{N' \cdot N^2} = R_{N'} + \frac{\epsilon}{N' \cdot N^2}$. It follows that R_N and $R_{N'}$ are almost equal:

$$R_{N'} < R_N < R_{N'} + \frac{1}{N^2}$$

Since this holds for every choice of c compromised tags, it also holds for the average case. The generalized upperbound for $1 \leq N < N'$ easily follows. □