Optimal Key-Trees for Tree-Based Private Authentication

Levente Buttyán, Tamás Holczer, and István Vajda

Laboratory of Cryptography and System Security (CrySyS)
Department of Telecommunications
Budapest University of Technology and Economics, Hungary
{`buttyan, holczer, vajda`}@crysys.hu

Abstract. Key-tree based private authentication has been proposed by
Molnar and Wagner as a neat way to efficiently solve the problem of pri-
vacy preserving authentication based on symmetric key cryptography.
However, in the key-tree based approach, the level of privacy provided
by the system to its members may decrease considerably if some mem-
bers are compromised. In this paper, we analyze this problem, and show
that careful design of the tree can help to minimize this loss of privacy.
First, we introduce a benchmark metric for measuring the resistance of
the system to a single compromised member. This metric is based on the
well-known concept of anonymity sets. Then, we show how the parame-
ters of the key-tree should be chosen in order to maximize the system's
resistance to single member compromise under some constraints on the
authentication delay. In the general case, when any member can be com-
promised, we give a lower bound on the level of privacy provided by
the system. We also present some simulation results that show that this
lower bound is quite sharp. The results of this paper can be directly used
by system designers to construct optimal key-trees in practice; indeed,
we consider this as the main contribution of our work.

1 Introduction

Entity authentication is the process whereby a party (the prover) corroborates
its identity to another party (the verifier). Entity authentication is often based
on authentication protocols in which the parties pass messages to each other.
These protocols are engineered in such a way that they resist various types
of impersonation and replay attacks [2]. However, less attention is paid to the
requirement of preserving the privacy of the parties (typically that of the prover)
with respect to an eavesdropping third party. Indeed, in many of the well-known
and widely used authentication protocols (e.g., [8, 10]) the identity of the prover
is sent in cleartext, and hence, it is revealed to an eavesdropper.

One approach to solve this problem is based on public key cryptography, and
it consists of encrypting the identity information of the prover with the public
key of the verifier so that no one but the verifier can learn the prover's iden-
tity. Another approach, also based on public key techniques, is that the parties

first run an anonymous Diffie-Hellman key exchange and establish a confidential channel, through which the prover can send its identity and authentication information to the verifier in a second step. An example for this second approach is the main mode of the Internet Key Exchange (IKE) protocol [7]. While it is possible to hide the identity of the prover by using the above mentioned approaches, they provide appropriate solution to the problem only if the parties can afford public key cryptography. In many applications, such as low cost RFID tags and contactless smart card based automated fare collection systems in mass transportation, this is not the case, while at the same time, the provision of privacy (especially location privacy) in those systems is strongly desirable.

The problem of using symmetric key encryption to hide the identity of the prover is that the verifier does not know which symmetric key it should use to decrypt the encrypted identity, because the appropriate key cannot be retrieved without the identity. The verifier may try all possible keys in its key database until one of them properly decrypts the encrypted identity[1], but this would increase the authentication delay if the number of potential provers is large. Long authentication delays are usually not desirable, moreover, in some cases, they may not even be acceptable. As an example, let us consider again contactless smart card based electronic tickets in public transportation: the number of smart cards in the system (i.e., the number of potential provers) may be very large in big cities, while the time needed to authenticate a card should be short in order to ensure a high throughput of passengers and avoid long queues at entry points.

Recently, Molnar and Wagner proposed an elegant approach to privacy protecting authentication [11] that is based on symmetric key cryptography while still ensuring short authentication delays. More precisely, the complexity of the authentication procedure in the Molnar-Wagner scheme is logarithmic in the number of potential provers, in contrast with the linear complexity of the naïve key search approach. The main idea of Molnar and Wagner is to use key-trees (see Figure 1 for illustration). A key-tree is a tree where a unique key is assigned to each edge. The leaves of the tree represent the potential provers, which we will call members in the sequel. Each member possesses the keys assigned to the edges of the path starting from the root and ending in the leaf that corresponds to the given member. The verifier knows all keys in the tree. In order to authenticate itself, a member uses all of its keys, one after the other, starting from the first level of the tree and proceeding towards lower levels. The verifier first determines which first level key has been used. For this, it needs to search through the first level keys only. Once the first key is identified, the verifier continues by determining which second level key has been used. However, for this, it needs to search through those second level keys only that reside below the already identified first level key in the tree. This process is continued until all keys are identified, which at the end, identify the authenticating member. The key point is that the verifier can reduce the search space considerably each time

---

[1] This of course requires redundancy in the encrypted message so that the verifier can determine if the decryption was successful.

a key is identified, because it should consider only the subtree below the recently identified key.
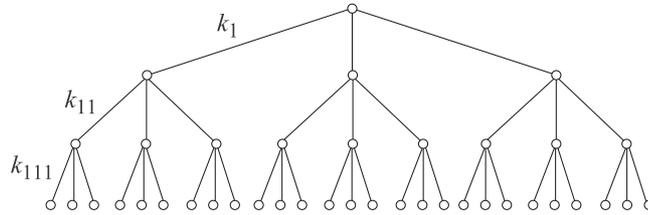
Fig. 1. Illustration of a key-tree. There is a unique key assigned to each edge. Each leaf represents a member of the system that possesses the keys assigned to the edges of the path starting from the root and ending in the given leaf. For instance, the member that belongs to the leftmost leaf in the figure possesses the keys k_1, k_{11}, and k_{111}.

The problem of the above described tree-based approach is that upper level keys in the tree are used by many members, and therefore, if a member is compromised and its keys become known to the adversary, then the adversary gains partial knowledge of the key of other members too [1]. This obviously reduces the privacy provided by the system to its members, since by observing the authentication of an uncompromised member, the adversary can recognize the usage of some compromised keys, and therefore its uncertainty regarding the identity of the authenticating member is reduced (it may be able to determine which subtree the member belongs to).

One interesting observation is that the naïve, linear key search approach can be viewed as a special case of the key-tree based approach, where the key-tree has a single level and each member has a single key. Regarding the above described problem of compromised members, the naïve approach is in fact optimal, because compromising a member does not reveal any key information of other members. At the same time, as we saw above, the authentication delay is the worst in this case. On the other hand, in case of a binary key-tree, we can observe that the compromise of a single member *strongly*[2] affects the privacy of the other members, while at the same time, the binary tree is very advantageous in terms of authentication delay. Thus, there seems to be a trade-off between the level of privacy provided by the system and the authentication delay, which depends on the parameters of the key-tree, but it is far from obvious to see how the optimal key-tree should look like. In this paper, we address this problem, and we show how to find optimal key-trees. More precisely, our main contributions are the following:

---

[2] The precise quantification of this effect is the topic of this paper and will be presented later.

- We propose a benchmark metric for measuring the resistance of the system to a single compromised member based on the concept of anonymity sets. To the best of our knowledge, anonymity sets have not been used in the context of key-tree based private authentication yet.
- We introduce the idea of using different branching factors at different levels of the key-tree; the advantage is that the system's resistance to single member compromise can be increased while still keeping the authentication delay short. To the best of our knowledge, key-trees with variable branching factors have not been proposed yet for private authentication.
- We present an algorithm for determining the optimal parameters of the key-tree, where optimal means that resistance to single member compromise is maximized, while the authentication delay is kept below a predefined threshold.
- In the general case, when any member can be compromised, we give a lower bound on the level of privacy provided by the system, and present some simulation results that show that this lower bound is quite sharp. This allows us to compare different systems based on their lower bounds.
- In summary, we propose *practically usable techniques* for designers of key-tree based authentication systems.

The outline of the paper is the following: In Section 2, we introduce our benchmark metric to measure the level of privacy provided by key-tree based authentication systems, and we illustrate, through an example, how this metric can be used to compare systems with different parameters. By the same token, we also show that key-trees with variable branching factors can be better than key-trees with a constant branching factor at every level. In Section 3, we formulate the problem of finding the best key-tree with respect to our benchmark metric as an optimization problem, and we present an algorithm that solves that optimization problem. In Section 4, we consider the general case, when any number of members can be compromised, and we derive a useful lower bound on the level of privacy provided by the system. Finally, in Section 5, we report on some related work, and in Section 6, we conclude the paper.

2 Resistance to single member compromise

There are different ways to measure the level of anonymity provided by a system [5, 14]. Here we will use the concept of anonymity sets [4]. The anonymity set of a member v is the set of members that are indistinguishable from v from the adversary's point of view. The size of the anonymity set is a good measure of the level of privacy provided for v, because it is related to the level of uncertainty of the adversary. Clearly, the larger the anonymity set is, the higher the level of privacy is. The minimum size of the anonymity set is 1, and its maximum size is equal to the number of all members in the system. In order to make the privacy measure independent of the number of members, one can divide the anonymity set size by the total number of members, and obtain a normalized

privacy measure between 0 and 1. Such normalization makes the comparison of different systems easier.

Now, let us consider a key-tree with ℓ levels and branching factors $b_1, b_2, \ldots, b_\ell$ at the levels, and let us assume that exactly one member is compromised (see Figure 2 for illustration). Knowledge of the compromised keys allows the adversary to partition the members into partitions $P_0, P_1, P_2, \ldots$, where

- P_0 contains the compromised member only,
- P_1 contains the members the parent of which is the same as that of the compromised member, and that are not in P_0,
- P_2 contains the members the grandparent of which is the same as that of the compromised member, and that are not in $P_0 \cup P_1$,
- etc.

Members of a given partition are indistinguishable for the adversary, while it can distinguish between members that belong to different partitions. Hence, each partition is the anonymity set of its members.
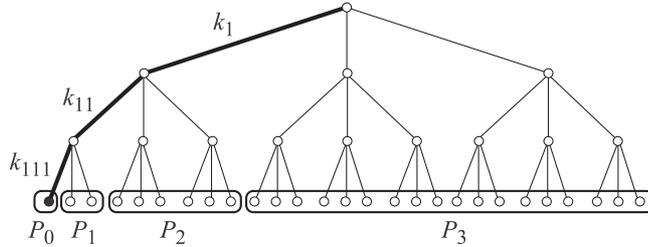
Fig. 2. Illustration of what happens when a single member is compromised. Without loss of generality, we assume that the member corresponding to the leftmost leaf in the figure is compromised. This means that the keys k_1, k_{11}, and k_{111} become known to the adversary. This knowledge of the adversary partitions the set of members into anonymity sets P_0, P_1, ... of different sizes. Members that belong to the same partition are indistinguishable to the adversary, while it can distinguish between members that belong to different partitions. For instance, the adversary can recognize a member in partition P_1 by observing the usage of k_1 and k_{11} but not that of k_{111}, where each of these keys are known to the adversary. Members in P_3 are recognized by not being able to observe the usage of any of the keys known to the adversary.

The level of privacy provided by the system can be characterized by the level of privacy provided to a randomly selected member, or in other words, by the expected size of the anonymity set of a randomly selected member. By definition, the expected anonymity set size is:

$$\bar{S} = \sum_{i=0}^{\ell} \frac{|P_i|}{N} |P_i| = \sum_{i=0}^{\ell} \frac{|P_i|^2}{N} \tag{1}$$

where N is the total number of members, and $|P_i|/N$ is the probability of selecting a member from partition P_i. We define the *resistance to single member compromise*, denoted by R, as the normalized expected anonymity set size, which can be computed as follows:

$$R = \frac{\bar{S}}{N} = \sum_{i=0}^{\ell} \frac{|P_i|^2}{N^2}$$
$$= \frac{1}{N^2} \left(1 + (b_\ell - 1)^2 + ((b_{\ell-1} - 1)b_\ell)^2 + \ldots + ((b_1 - 1)b_2 b_3 \ldots b_\ell)^2 \right)$$
$$= \frac{1}{N^2} \left(1 + (b_\ell - 1)^2 + \sum_{i=1}^{\ell-1} (b_i - 1)^2 \prod_{j=i+1}^{\ell} b_j^2 \right) \qquad (2)$$

where we used that

$$|P_0| = 1$$
$$|P_1| = b_\ell - 1$$
$$|P_2| = (b_{\ell-1} - 1)b_\ell$$
$$|P_3| = (b_{\ell-2} - 1)b_{\ell-1}b_\ell$$
$$\ldots \quad \ldots$$
$$|P_\ell| = (b_1 - 1)b_2 b_3 \ldots b_\ell$$

As its name indicates, R characterizes the loss of privacy due to the compromise of a single member of the system. If R is close to 1, then the expected anonymity set size is close to the total number of members, and hence, the loss of privacy is small. On the other hand, if R is close to 0, then the loss of privacy is high, as the expected anonymity set size is small. We use R as a benchmark metric based on which different systems can be compared.

Obviously, a system with greater R is better, and therefore, we would like to maximize R. However, there are some constraints. We define the *maximum authentication delay*, denoted by D, as the number of basic operations needed to authenticate any member in the worst case. The maximum authentication delay in case of key-tree based authentication can be computed as $D = \sum_{i=1}^{\ell} b_i$. In most practical cases, there is an upper bound D_{max} on the maximum authentication delay allowed in the system. For instance, in the specification for electronic ticketing systems for public transport applications in Hungary [6], it is required that a ticket validation transaction should be completed in 250 ms. Taking into account the details of the ticket validation protocol, one can derive D_{max} for electronic tickets from such specifications. Therefore, in practice, the designer's task is to maximize R under the constraint that $D \le D_{max}$. We will address this problem in Section 3.

In the remainder of this section, we illustrate how the benchmark metric R can be used to compare different systems. This exercise will also lead to an important revelation: key-trees with varying branching factors at different levels

could provide higher level of privacy than key-trees with a constant branching factor, while having the same or even a shorter authentication delay.

Example: Let us assume that the total number N of members is 27000 and the upper bound D_{max} on the maximum authentication delay is 90. Let us consider a key-tree with a constant branching factor vector $B = (30, 30, 30)$, and another key-tree with branching factor vector $B' = (60, 10, 9, 5)$. Both key-trees can serve the given population of members, since $30^3 = 60 \cdot 10 \cdot 9 \cdot 5 = 27000$. In addition, both key-trees ensure that the maximum authentication delay is not longer than D_{max}: for the first key-tree, we have $D = 3 \cdot 30 = 90$, whereas for the second one, we get $D = 60+10+9+5 = 84$. Using (2), we can compute the resistance to single member compromise for both key-trees. For the first tree, we get $R \approx 0.9355$, while for the second tree we obtain $R \approx 0.9672$. Thus, we arrive to the conclusion that the second key-tree with variable branching factors is better, as it provides a higher level of privacy, while ensuring a smaller authentication delay.

At this point, several questions arise naturally: Is there an even better branching factor vector than B' for $N = 27000$ and $D_{max} = 90$? What is the best branching factor vector for this case? How can we find the best branching factor vector in general? We give the answers to these questions in the next section.

3   Optimal trees in case of single member compromise

The problem of finding the best branching factor vector can be described as an optimization problem as follows: *Given the total number N of members and the upper bound D_{max} on the maximum authentication delay, find a branching factor vector $B = (b_1, b_2, \ldots b_\ell)$ such that $R(B)$ is maximal subject to the following constraints:*

$$\prod_{i=1}^{\ell} b_i = N \tag{3}$$

$$\sum_{i=1}^{\ell} b_i \leq D_{max} \tag{4}$$

We analyze this optimization problem through a series of lemmas that will lead to an algorithm that solves the problem. Our first lemma states that we can always improve a branching factor vector by ordering its elements in decreasing order, and hence, in the sequel we will consider only ordered vectors:

Lemma 1. *Let N and D_{max} be the total number of members and the upper bound on the maximum authentication delay, respectively. Moreover, 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 the optimization problem defined above, then B^* also satisfies them, and $R(B^*) \geq R(B)$.*

Proof. The proof can be found in the Appendix.

The following lemma provides a lower bound and an upper bound for the resistance to single member compromise:

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

$$\left(1 - \frac{1}{b_1}\right)^2 \leq R(B) \leq \left(1 - \frac{1}{b_1}\right)^2 + \frac{4}{3b_1^2} \tag{5}$$

Proof. The proof can be found in the Appendix.

Let us consider the bounds in Lemma 2. Note that the branching factor vector is ordered, therefore, b_1 is not smaller than any other b_i. We can observe that if we increase b_1, then the difference between the upper and the lower bounds decreases, and $R(B)$ gets closer to 1. Intuitively, this implies that in order to find the solution to the optimization problem, b_1 should be maximized. The following lemma underpins this intuition formally:

Lemma 3. *Let N and D_{max} be the total number of members and the upper bound on the maximum authentication delay, respectively. Moreover, let $B = (b_1, b_2, \ldots, b_\ell)$ and $B' = (b'_1, b'_2, \ldots, b'_{\ell'})$ be two sorted branching factor vectors that satisfy the constraints of the optimization problem defined above. Then, $b_1 > b'_1$ implies $R(B) \geq R(B')$.*

Proof. The proof can be found in the Appendix.

Lemma 3 states that given two branching factor vectors, the one with the larger first element is always at least as good as the other. The next lemma generalizes this result 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.

Lemma 4. *Let N and D_{max} be the total number of members and the upper bound on the maximum authentication delay, respectively. Moreover, let $B = (b_1, b_2, \ldots, b_\ell)$ and $B' = (b'_1, b'_2, \ldots, b'_{\ell'})$ be two sorted branching factor vectors such that $b_i = b'_i$ for all $1 \leq i \leq j$ for some $j < \min(\ell, \ell')$, and both B and B' satisfy the constraints of the optimization problem defined above. Then, $b_{j+1} > b'_{j+1}$ implies $R(B) \geq R(B')$.*

Proof. The proof can be found in the Appendix.

We will now present an algorithm that finds the solution to the optimization problem. However, before doing that, we need to introduce some further notations. Let $B = (b_1, b_2, \ldots, b_\ell)$ and $B' = (b'_1, b'_2, \ldots, b'_{\ell'})$. Then

- $\prod(B)$ denotes $\prod_{i=1}^{\ell} b_i$;
- $\sum(B)$ denotes $\sum_{i=1}^{\ell} b_i$;
- $\{B\}$ denotes the set $\{b_1, b_2, \ldots, b_\ell\}$ of the elements of B;

- $B' \subseteq B$ means that $\{B'\} \subseteq \{B\}$;
- if $B' \subseteq B$, then $B \setminus B'$ denotes the vector that consists of the elements of $\{B\} \setminus \{B'\}$ in decreasing order;
- if b is a positive integer, then $b|B$ denotes the vector $(b, b_1, b_2, \ldots, b_\ell)$.

We define our algorithm as a recursive function f, which takes two input parameters, a vector B of positive integers, and another positive integer d, and returns a vector of positive integers. In order to compute the optimal branching factor vector for a given N and D_{max}, f should be called with the vector that contains the prime factors of N, and D_{max}. For instance, if $N = 27000$ and $D_{max} = 90$ (we use the same parameters as in the example in Sec 2, to compare the naïve and algorithmical results), then f should be called with $B = (5, 5, 5, 3, 3, 3, 2, 2, 2)$ and $d = 90$. Function f will then return the optimal branching factor vector.

Function f is defined as follows:

$f(B, d)$
1   if $\sum(B) > d$ then exit (no solution exists)
2   else find $B' \subseteq B$ such that
         $\prod(B') + \sum(B \setminus B') \leq d$ and
         $\prod(B')$ is maximal
3   if $B' = B$ then return $(\prod(B'))$
4   else return $\prod(B')|f(B \setminus B', d - \prod(B'))$

The operation of the algorithm can be described as follows: The algorithm starts with a branching factor vector consisting of the prime factors of N. This vector satisfies the first constraint of the optimization problem by definition. If it does not satisfy the second constraint (i.e., it does not respect the upper bound on the maximum authentication delay), then no solution exists. Otherwise, the algorithm successively improves the branching factor vector by maximizing its elements, starting with the first element, and then proceeding to the next elements, one after the other. Maximization of an element is done by joining as yet unused prime factors until the resulting divisor of N cannot be further increased without violating the constraints of the optimization problem.

Theorem 1. *Let N and D_{max} be the total number of members and the upper bound on the maximum authentication delay, respectively. Moreover, let B be a vector that contains the prime factors of N. Then, $f(B, D_{max})$ is an optimal branching factor vector for N and D_{max}.*

Proof. We will give a sketch of the proof. Let $B^* = f(B, D_{max})$, and let us assume that there is another branching factor vector $B' \neq B^*$ that also satisfies the constraints of the optimization problem and $R(B') > R(B^*)$. We will show that this leads to a contradiction, hence B^* should be optimal.

Let $B^* = (b_1^*, b_2^*, \ldots, b_{\ell^*}^*)$ and $B' = (b_1', b_2', \ldots, b_{\ell'}')$. Recall that B^* is obtained by first maximizing the first element in the vector, therefore, $b_1^* \geq b_1'$ must hold. If $b_1^* > b_1'$, then $R(B^*) \geq R(B')$ by Lemma 3, and thus, B' cannot be a better vector than B^*. This means that $b_1^* = b_1'$ must hold.

We know that once b_1^* is determined, our algorithm continues by maximizing the next element of B^*. Hence, $b_2^* \geq b_2'$ must hold. If $b_2^* > b_2'$, then $R(B^*) \geq R(B')$ by Lemma 4, and thus, B' cannot be a better vector than B^*. This means that $b_2^* = b_2'$ must hold too.

By repeating this argument, finally, we arrive to the conclusion that $B^* = B'$ must hold, which is a contradiction. $\diamond$

Table 1 illustrates the operation of the algorithm for $B = (5, 5, 5, 3, 3, 3, 2, 2, 2)$ and $d = 90$. The rows of the table correspond to the levels of the recursion during the execution. The column labelled with B' contains the prime factors that are joined at a given recursion level. The optimal branching factor vector can be read out from the last column of the table (each row contains one element of the vector). From this example, we can see that the optimal branching factor vector for $N = 27000$ and $D_{max} = 90$ is $B^* = (72, 5, 5, 5, 3)$. For the key-tree defined by this vector, we get $R \approx 0.9725$, and $D = 90$.

recursion level	B	d	B'	$\prod(B')$
1	$(5, 5, 5, 3, 3, 3, 2, 2, 2)$	90	$(3, 3, 2, 2, 2)$	72
2	$(5, 5, 5, 3)$	18	(5)	5
3	$(5, 5, 3)$	13	(5)	5
4	$(5, 3)$	8	(5)	5
5	(3)	3	(3)	3

Table 1. Illustration of the operation of the recursive function f when called with $B = (5, 5, 5, 3, 3, 3, 2, 2, 2)$ and $d = 90$. The rows of the table correspond to the levels of the recursion during the execution.

4 Analysis of the general case

So far, we have studied the case of a single compromised member. This already proved to be useful, because it allowed us to compare different key-trees and to derive a key-tree construction method. However, one may still be interested in what level of privacy is provided by a system in the general case when any number of members could be compromised. In this section, we address this problem.

In what follows, we will need to refer to the non-leaf vertices of the key-tree, and for this reason, we introduce the labelling scheme that is illustrated in Figure 3. In addition, we need to introduce some further notations. We call a leaf compromised if it belongs to a compromised member, and we call a non-leaf vertex compromised if it lies on a path that leads to a compromised leaf in the tree. If vertex v is compromised, then

- K_v denotes the set of the compromised children of v, and $k_v = |K_v|$;
- $\mathcal{P}_v$ denotes the set of partitions (anonymity sets) that belong to the subtree rooted at v (see Figure 3 for illustration); and
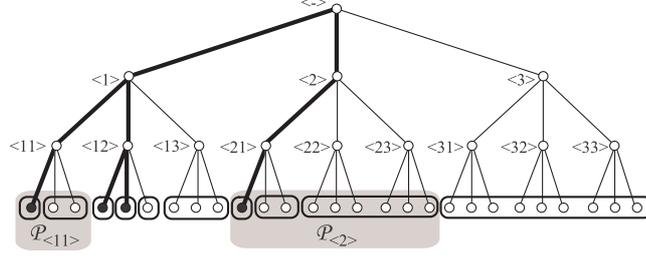
Fig. 3. Illustration of what happens when several members are compromised. Just as in the case of a single compromised member, the members are partitioned into anonymity sets, but now the resulting partitions depend on the number of the compromised members, as well as on their positions in the tree. Nevertheless, the expected size of the anonymity set of a randomly selected member is still a good metric for the level of privacy provided by the system, although, in this general case, it is more difficult to compute.

- $\bar{S}_v$ denotes the average size of the partitions in $\mathcal{P}_v$.

We are interested in computing $\bar{S}_{\langle - \rangle}$. We can do that as follows:

$$
\begin{aligned}
\bar{S}_{\langle - \rangle} &= \sum_{P \in \mathcal{P}_{\langle - \rangle}} \frac{|P|^2}{b_1 b_2 \dots b_\ell} \\
&= \frac{((b_1 - k_{\langle - \rangle}) b_2 \dots b_\ell)^2}{b_1 b_2 \dots b_\ell} + \sum_{v \in K_{\langle - \rangle}} \sum_{P \in \mathcal{P}_v} \frac{|P|^2}{b_1 b_2 \dots b_\ell} \\
&= \frac{((b_1 - k_{\langle - \rangle}) b_2 \dots b_\ell)^2}{b_1 b_2 \dots b_\ell} + \frac{1}{b_1} \sum_{v \in K_{\langle - \rangle}} \bar{S}_v
\end{aligned}
\tag{6}
$$

In general, for any vertex $\langle i_1, \dots, i_j \rangle$ such that $1 \le j < \ell - 1$:

$$
\bar{S}_{\langle i_1, \dots, i_j \rangle} = \frac{((b_{j+1} - k_{\langle i_1, \dots, i_j \rangle}) b_{j+2} \dots b_\ell)^2}{b_{j+1} \dots b_\ell} + \frac{1}{b_{j+1}} \sum_{v \in K_{\langle i_1, \dots, i_j \rangle}} \bar{S}_v
\tag{7}
$$

Finally, for vertices $\langle i_1, \dots, i_{\ell-1} \rangle$ just above the leaves, we get:

$$
\bar{S}_{\langle i_1, \dots, i_{\ell-1} \rangle} = \frac{(b_\ell - k_{\langle i_1, \dots, i_{\ell-1} \rangle})^2}{b_\ell} + \frac{k_{\langle i_1, \dots, i_{\ell-1} \rangle}}{b_\ell}
\tag{8}
$$

Expressions (6 – 8) can be used to compute the expected anonymity set size in the system iteratively, in case of any number of compromised members. However, note that the computation depends not only on the number c of the

compromised members, but also their positions in the tree. This makes the comparison of different systems difficult, because for a comprehensive analysis, all possible allocations of the compromised members over the leaves of the key-tree should be considered. Therefore, we would prefer a formula that depends solely on c, but characterizes the effect of compromised members on the level of privacy sufficiently well, so that it can serve as a basis for comparison of different systems. In the following, we derive such a formula based on the assumption that the compromised members are distributed uniformly at random over the leaves of the key-tree. In some sense, this is a pessimistic assumption as the uniform distribution represents the worst case, which leads to the largest amount of privacy loss due to the compromised members. Thus, the approximation that we derive can be viewed as a lower bound on the expected anonymity set size in the system when c members are compromised.

Let the branching factor of the key-tree be $B = (b_1, b_2, \ldots, b_\ell)$, and let c be the number of compromised leaves in the tree. We can estimate $k_{\langle - \rangle}$ for the root as follows:

$$k_{\langle - \rangle} \approx \min(c, b_1) = k_0 \tag{9}$$

If a vertex $\langle i \rangle$ at the first level of the tree is compromised, then the number of compromised leaves in the subtree rooted at $\langle i \rangle$ is approximately $c/k_0 = c_1$. Then, we can estimate $k_{\langle i \rangle}$ as follows:

$$k_{\langle i \rangle} \approx \min(c_1, b_2) = k_1 \tag{10}$$

In general, if vertex $\langle i_1, \ldots, i_j \rangle$ at the j-th level of the tree is compromised, then the number of compromised leaves in the subtree rooted at $\langle i_1, \ldots, i_j \rangle$ is approximately $c_{j-1}/k_{j-1} = c_j$, and we can use this to approximate $k_{\langle i_1, \ldots, i_j \rangle}$ as follows:

$$k_{\langle i_1, \ldots, i_j \rangle} \approx \min(c_j, b_{j+1}) = k_j \tag{11}$$

Using these approximations in expressions $(6 - 8)$, we can derive an approximation for $\bar{S}_{\langle - \rangle}$, which we denote by $\bar{S}_0$, in the following way:

$$\bar{S}_{\ell-1} = \frac{(b_\ell - k_{\ell-1})^2}{b_\ell} + \frac{k_{\ell-1}}{b_\ell} \tag{12}$$

$$\ldots \quad \ldots$$

$$\bar{S}_j = \frac{((b_{j+1} - k_j)b_{j+2} \ldots b_\ell)^2}{b_{j+1} \ldots b_\ell} + \frac{k_j}{b_{j+1}} \bar{S}_{j+1} \tag{13}$$

$$\ldots \quad \ldots$$

$$\bar{S}_0 = \frac{((b_1 - k_0)b_2 \ldots b_\ell)^2}{b_1 \ldots b_\ell} + \frac{k_0}{b_1} \bar{S}_1 \tag{14}$$

Note that expressions $(14 - 12)$ do not depend on the positions of the compromised leaves in the tree, but they depend only on the value of c.

In order to see how well $\bar{S}_0$ estimates $\bar{S}_{\langle - \rangle}$, we run some simulations. The simulation parameters were the following:

– total number of members $N = 27000$;

- upper bound on the maximum authentication delay $D_{max} = 90$;
- we considered two branching factor vectors: $(30, 30, 30)$ and $(72, 5, 5, 5, 3)$;
- we varied the number c of compromised members between 1 and 270 with a step size of one.

For each value of c, we run 100 simulations[3]. In each simulation run, the c compromised members were chosen uniformly at random from the set of all members. We computed the exact value of the normalized expected anonymity set size $\bar{S}_{\langle-\rangle}/N$ using the expressions $(6 - 8)$. Finally, we averaged the obtained values over all simulation runs. Moreover, for every c, we also computed the estimated value $\bar{S}_0/N$ using the expressions $(14 - 12)$.

The simulation results are shown in Figure 4. The figure does not show the confidence interwalls, because they are very small (in the range of 10^{-4} for all simulations) and thus they could be hardly visible. As we can see, $\bar{S}_0/N$ approximates $\bar{S}_{\langle-\rangle}/N$ quite well, and in general it provides a lower bound on the normalized expected anonymity set size.
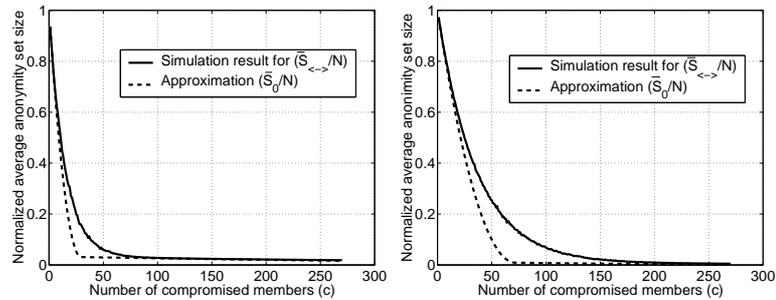
Fig. 4. Simulation results for branching factor vectors $(30, 30, 30)$ (left hand side) and $(72, 5, 5, 5, 3)$ (right hand side). As we can see, $\bar{S}_0/N$ approximates $\bar{S}_{\langle-\rangle}/N$ quite well, and in general it provides a lower bound on it.

In Figure 5, we plotted the value of $\bar{S}_0/N$ as a function of c for different branching factor vectors. This figure illustrates, how different systems can be compared using our approximation $\bar{S}_0/N$ of the normalized expected anonymity set size. On the left hand side of the figure, we can see that the value of $\bar{S}_0/N$ is greater for the vector $B^* = (72, 5, 5, 5, 3)$ than for the vector $B = (30, 30, 30)$ not only for $c = 1$ (as we saw before), but for larger values of c too. In fact, B^* seems to lose its superiority only when the value of c approaches 60, but at this range, the systems nearly provide no privacy in any case. Thus, we can conclude that B^* is a better branching factor vector yielding more privacy than B in general.

---

[3] All computations have been done in Matlab, and for the purpose of repeatability, the source code is available on-line at http://www.crysys.hu/~holczer/PET2006
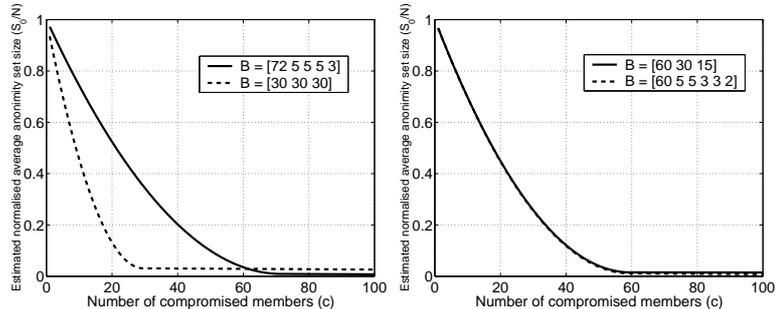
Fig. 5. The value of $\bar{S}_0/N$ as a function of c for different branching factor vectors. The figure illustrates, how different systems can be compared based on the approximation $\bar{S}_0/N$. On the left hand side, we can see that the value of $\bar{S}_0/N$ is greater for the vector $(72, 5, 5, 5, 3)$ than for the vector $(30, 30, 30)$ not only for $c = 1$ (as we saw earlier), but for larger values of c too. On the right hand side, we can see that $\bar{S}_0/N$ is almost the same for the vector $(60, 5, 5, 3, 3, 2)$ as for the vector $(60, 30, 15)$. We can conclude that $\bar{S}_0/N$ is essentially determined by the value of the first element of the branching factor vector.

We can make another interesting observation on the left hand side of Figure 5: $\bar{S}_0/N$ starts decreasing sharply as c starts increasing, however, when c gets close to the value of the first element of the branching factor vector, the decrease of $\bar{S}_0/N$ slows down. Moreover, almost exactly when c reaches the value of the first element (30 in case of B, and 72 in case of B^*), $\bar{S}_0/N$ seems to turn into constant, but at a very low value. We can conclude that, just as in the case of a single compromised member, in the general case too, the level of privacy provided by the system essentially depends on the value of the first element of the branching factor vector. The plot on the right hand side of the figure reinforces this observation: it shows $\bar{S}_0/N$ for two branching factor vectors that have the same first element but that differ in the other elements. As we can see, the curves are almost perfectly overlapping.

Thus, a practical design principle for key-tree based private authentication systems is to maximize the branching factor at the first level of the key-tree. Further optimization by adjusting the branching factors of the lower levels may still be possible, but the gain is not significant; what really counts is the branching factor at the first level.

5 Related work

The problem of private authentication has been extensively studied in the literature recently, but most of the proposed solutions are based on public key cryptography. One example is Idemix, which is a practical anonymous credential system proposed by Camenisch and Lysyanskaya in [3]. Idemix allows for unlinkable demonstration of the possession of various credentials, and it can be

used in many applications. However, it is not applicable in resource constraint scenarios, such as low-cost RFID systems. For such applications, solutions based on symmetric key cryptography seem to be the only viable options.

The key-tree based approach for symmetric key private authentication has been proposed by Molnar and Wagner in [11]. However, they use a simple b-ary tree, which means that the tree has the same branching factor at every level. Moreover, they do not analyze the effects of compromised members on the level of privacy provided. They only mention that compromise of a member has a wider effect than in the case of public key cryptography based solutions.

An entropy based analyzis of key trees can be found in [12]. Nohara *et al.* prove that their K-steps ID matching scheme (whitch is very similar to [11]) is secure against one compromised tag, if the number of nodes are large enough. They consider only b-ary trees, no variable branching factors. The entropy based analysis leads to a slightly different optimization problem. We leave the detailed comparison of the entropy based and the anonimity set based approach for future work.

Finally, Avoine *et al.* analyze the effects of compromised members on privacy in the key-tree based approach [1]. They study the case of a single compromised member, as well as the general case of any compromised members. However, their analysis is not based on the notion of anonymity sets. In their model, the adversary is first allowed to compromise some members, then it chooses a target member that it wants to trace, and it is allowed to interact with the chosen member. Later, the adversary is given two members such that one of them is the target member chosen by the adversary. The adversary can interact with the given members, and it must decide which one is its target. The level of privacy provided by the system is quantified by the success probability of the adversary. This model is similar to ours in case of a single compromised member, but it is slightly different in the general case. Moreover, Avoine *et al.* do not consider the problem of how to optimize the key-tree, instead, they suggest a time-memory trade-off to reduce the authentication delay.

6 Conclusion

Key-trees provide an efficient solution for private authentication in the symmetric key setting. However, the level of privacy provided by key-tree based systems decreases considerably if some members are compromised. The main message of this paper is that this loss of privacy can be minimized by the careful design of the tree. Based on our results presented in this paper, we can conclude that a good practical design principle is to maximize the branching factor at the first level of the tree such that the resulting tree still respects the constraint on the maximum authentication delay in the system. Once the branching factor at the first level is maximized, the tree can be further optimized by maximizing the branching factors at the successive levels, but the improvement achieved in this way is not really significant; what really counts is the branching factor at the first level.

Acknowledgements

References

1. G. Avoine, E. Dysli, and P. Oechslin. Reducing time complexity in RFID systems. In *Proceedings of the 12th Annual Workshop on Selected Areas in Cryptography (SAC'05)*, 2005.
2. C. Boyd, A. Mathuria. Protocols for Authentication and Key Establishment. Springer-Verlag, 2003
3. J. Camenisch, A. Lysyanskaya. A Efficient Non-transferable Anonymous Multi-show Credential System with Optional Anonymity Revocation. In *Advances in Cryptography – EUROCRYPT 2001*. Springer, 2001.
4. D. Chaum. The Dining Cryptographers Problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.
5. C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In Dingledine and Syverson (Eds.), *Designing Privacy Enhancing Technologies*, Springer LNCS 2482, pp. 54–68, 2002.
6. Elektra Hungaria (In Hungarian)
   http://www.gkm.gov.hu/data/357863/kovetelmeny1215.pdf
7. IKE, The Internet Key Exchange, RFC 2409, http://www.ietf.org/rfc/rfc2409.txt
8. ISO 9798-2. Mechanisms using symmetric encipherment algorithms http://www.iso.org
9. A. Juels. RFID security and privacy: a research survey. Manuscript, condensed version will appear in the *IEEE Journal on Selected Areas in Communication*, September 2005.
10. Kerberos. RFC 1510, http://www.ietf.org/rfc/rfc1510.txt
11. D. Molnar and D. Wagner. Privacy and security in library RFID: issues, practices, and architectures. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2004.
12. Y. Nohara, S. Inoue, K. Baba, H. Yasuura. Quantitative Evaluation of Unlinkable ID Matching Schemes. In Workshop on Privacy in the Electronic Society, WPES, 2005.
13. A. Pfitzmann and M. Khntopp. Anonymity, unobservability and pseudonymity – a proposal for terminology. In *Proceedings of the Privacy Enhancing Technologies (PET) Workshop*, Springer LNCS 2009, pp. 1–9, 2001.
14. A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of the Privacy Enhancing Technologies (PET) Workshop*, Springer LNCS, 2002.

A Proof of Lemma 1

B^* has the same elements as B has, therefore, the sum and the product of the elements of B^* are the same as that of B, and so if B satisfies the constraints of the optimization problem, then B^* does so too.

Now, let us assume that B^* is obtained from B with the bubble sort algorithm. The basic step of this algorithm is to change two neighboring elements if they are not in the right order. Let us suppose that $b_i < b_{i+1}$, and thus, the algorithm changes the order of b_i and b_{i+1}. Then, using (2), we can express $\Delta R = R(B^*) - R(B)$ as follows:

$$\Delta R = \frac{1}{N^2} \left((b_{i+1} - 1)^2 b_i^2 \prod_{j=i+2}^{\ell} b_j^2 + (b_i - 1)^2 \prod_{j=i+2}^{\ell} b_j^2 \right) -$$

$$\frac{1}{N^2} \left((b_i - 1)^2 b_{i+1}^2 \prod_{j=i+2}^{\ell} b_j^2 + (b_{i+1} - 1)^2 \prod_{j=i+2}^{\ell} b_j^2 \right)$$

$$= \frac{\prod_{j=i+2}^{\ell} b_j^2}{N^2} \left((b_{i+1} - 1)^2 b_i^2 + (b_i - 1)^2 - (b_i - 1)^2 b_{i+1}^2 - (b_{i+1} - 1)^2 \right)$$

$$= \frac{\prod_{j=i+2}^{\ell} b_j^2}{N^2} \left((b_{i+1} - 1)^2 (b_i^2 - 1) - (b_i - 1)^2 (b_{i+1}^2 - 1) \right)$$

$$= \frac{(b_i - 1)(b_{i+1} - 1) \prod_{j=i+2}^{\ell} b_j^2}{N^2} \left((b_{i+1} - 1)(b_i + 1) - (b_i - 1)(b_{i+1} + 1) \right)$$

Since $b_i \geq 2$ for all i, ΔR is non-negative if

$$\frac{b_i + 1}{b_i - 1} \geq \frac{b_{i+1} + 1}{b_{i+1} - 1} \tag{15}$$

But (15) must hold, since the function $f(x) = \frac{x+1}{x-1}$ is a monotone decreasing function, and by assumption, $b_i < b_{i+1}$. This means, that when sorting the elements of B, we improve $R(B)$ in every step, and thus, $R(B^*) \geq R(B)$ must hold. $\diamond$

B   Proof of Lemma 2

By definition

$$R = \frac{1}{N^2} \left(1 + (b_\ell - 1)^2 + \sum_{i=1}^{\ell-1} (b_i - 1)^2 \prod_{j=i+1}^{\ell} b_j^2 \right)$$

$$= \left(\frac{b_1 - 1}{b_1} \right)^2 + \frac{1}{N^2} \left(1 + (b_\ell - 1)^2 + \sum_{i=2}^{\ell-1} (b_i - 1)^2 \prod_{j=i+1}^{\ell} b_j^2 \right) \tag{16}$$

where we used that $N = b_1 b_2 \ldots b_\ell$. The lower bound in the lemma[4] follows directly from (16). In order to obtain the upper bound, we write b_i instead of

---

[4] Note that we could also derive the slightly better lower bound of $\left(\frac{b_1 - 1}{b_1} \right)^2 + \frac{1}{N^2}$ from (16), however, we do not need that in this paper.

$(b_i - 1)$ in the sum in (16):

$$R < \left(\frac{b_1 - 1}{b_1}\right)^2 + \frac{1}{N^2}\left(1 + \sum_{i=2}^{\ell}\prod_{j=i}^{\ell} b_j^2\right)$$

$$= \left(\frac{b_1 - 1}{b_1}\right)^2 + \frac{1}{b_1^2}\left(1 + \sum_{i=2}^{\ell}\prod_{j=2}^{i}\frac{1}{b_j^2}\right)$$

Since $b_i \geq 2$ for all i, we can write 2 in place of b_i in the sum, and we obtain:

$$R < \left(\frac{b_1 - 1}{b_1}\right)^2 + \frac{1}{b_1^2}\left(1 + \sum_{i=2}^{\ell}\prod_{j=2}^{i}\frac{1}{4}\right)$$

$$= \left(\frac{b_1 - 1}{b_1}\right)^2 + \frac{1}{b_1^2}\left(1 + \sum_{i=2}^{\ell}\left(\frac{1}{4}\right)^{i-1}\right)$$

$$< \left(\frac{b_1 - 1}{b_1}\right)^2 + \frac{1}{b_1^2}\left(1 + \sum_{i=2}^{\infty}\left(\frac{1}{4}\right)^{i-1}\right)$$

$$= \left(\frac{b_1 - 1}{b_1}\right)^2 + \frac{1}{b_1^2}\frac{1}{1 - \frac{1}{4}}$$

and this is the upper bound in the lemma. $\diamond$

C   Proof of Lemma 3

First, we prove that the statement of the lemma is true if $b_1' \geq 5$. We know from Lemma 2 that

$$R(B') < \left(1 - \frac{1}{b_1'}\right)^2 + \frac{4}{3b_1'^2}$$

and

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

where we used that $b_1 > b_1'$ by assumption. If we can prove that

$$\left(1 - \frac{1}{b_1'}\right)^2 + \frac{4}{3b_1'^2} \leq \left(1 - \frac{1}{b_1' + 1}\right)^2 \tag{17}$$

then we also proved that $R(B') \leq R(B)$. Indeed, a straightforward calculation yields that (17) is true if $b_1' \geq 2 + \sqrt{\frac{15}{2}}$, and since b_1' is an integer, we are done.

Next, we make the observation that a branching factor vector $A = (a_1, \ldots, a_k, 2, 2)$ that has at least two 2s at the end can be improved by joining two 2s into a 4 and obtaining $A' = (a_1, \ldots, a_k, 4)$. It is clear that neither the sum nor the

product of the elements changes with this transformation. In addition, we can use the definition of R to get

$$N^2 \cdot R(A) = ((a_1 - 1) \cdot a_2 \cdot \ldots \cdot a_k \cdot 2 \cdot 2)^2 + \ldots + ((a_k - 1) \cdot 2 \cdot 2)^2 +$$
$$((2-1) \cdot 2)^2 + (2-1)^2 + 1$$

and

$$N^2 \cdot R(A') = ((a_1 - 1) \cdot a_2 \cdot \ldots \cdot a_k \cdot 4)^2 + \ldots + ((a_k - 1) \cdot 4)^2 +$$
$$(4-1)^2 + 1$$

Thus, $R(A') - R(A) = \frac{1}{N^2}(9 - 4 - 1) > 0$, which means that A' is better than A.

Now, we prove that the lemma is also true for $b'_1 \in \{2, 3, 4\}$:

- $b'_1 = 2$: Since B' is an ordered vector where b'_1 is the largest element, it follows that every element of B' is 2, and thus, N is a power of 2. From Lemma 2, $R(B') < (1 - \frac{1}{2})^2 + \frac{4}{3 \cdot 2^2} = \frac{7}{12}$ and $R(B) > (1 - \frac{1}{b_1})^2$. It is easy to see that $(1 - \frac{1}{b_1})^2 \geq \frac{7}{12}$ if $b_1 \geq \frac{1}{1 - \sqrt{\frac{7}{12}}} = 4.23$. Since $b_1 > b'_1$, the remaining cases are $b_1 = 3$ and $b_1 = 4$. However, $b_1 = 3$ cannot be the case, because N is a power of 2. If $b_1 = 4$, then B can be obtained from B' by joining pairs of 2s into 4s and then ordering the elements. However, according to our observation above and Lemma 1, both operations improve the vector. It follows that $R(B) \geq R(B')$ must hold.
- $b'_1 = 3$: From Lemma 2, $R(B') < (1 - \frac{1}{3})^2 + \frac{4}{3 \cdot 3^2} = \frac{16}{27}$ and $R(B) > (1 - \frac{1}{b_1})^2$. It is easy to see that $(1 - \frac{1}{b_1})^2 \geq \frac{16}{27}$ if $b_1 \geq \frac{9}{9 - 4 \cdot \sqrt{3}} = 4.34$. Since $b_1 > b'_1$, the only remaining case is $b_1 = 4$. In this case, the vectors are as follows:

$$B = (\overbrace{2^2, \ldots, 2^2}^{i}, \overbrace{3, \ldots, 3}^{j}, \overbrace{2, \ldots, 2}^{k})$$

$$B' = (\overbrace{3, \ldots, 3}^{j}, \overbrace{2, \ldots, 2}^{2i+k})$$

where $i, j \geq 1$ and $k \geq 0$. This means that B can be obtained from B' by joining i pairs of 2s into 4s and then ordering the elements. However, as we saw earlier, both joining 2s into 4s and ordering the elements improve the vector, and thus, $R(B) \geq R(B')$ must hold.
- $b'_1 = 4$: Since B' is an ordered vector where b'_1 is the largest element, it follows that N is not divisible by 5. From Lemma 2, $R(B') < (1 - \frac{1}{4})^2 + \frac{4}{3 \cdot 4^2} = \frac{31}{48}$ and $R(B) > (1 - \frac{1}{b_1})^2$. It is easy to see that $(1 - \frac{1}{b_1})^2 \geq \frac{31}{48}$ if $b_1 \geq \frac{1}{1 - \sqrt{\frac{31}{48}}} = 5.09$. Since $b_1 > b'_1$, the remaining case is $b_1 = 5$. However, $b_1 = 5$ cannot be the case, because N is not divisible by 5. $\diamond$

D Proof of Lemma 4

By definition

$$R(B) = \frac{1}{N^2} \left(1 + (b_\ell - 1)^2 + \sum_{i=1}^{\ell-1} (b_i - 1)^2 \prod_{j=i+1}^{\ell} b_j^2 \right)$$

$$= \left(\frac{b_1 - 1}{b_1} \right)^2 + \frac{1}{b_1^2} \left(\frac{1}{(N/b_1)^2} \left(1 + (b_\ell - 1)^2 + \sum_{i=2}^{\ell-1} (b_i - 1)^2 \prod_{j=i+1}^{\ell} b_j^2 \right) \right)$$

$$= \left(\frac{b_1 - 1}{b_1} \right)^2 + \frac{1}{b_1^2} \cdot R(B_1)$$

where $B_1 = (b_2, b_3, \ldots, b_\ell)$. Similarly,

$$R(B') = \left(\frac{b_1' - 1}{b_1'} \right)^2 + \frac{1}{b_1'^2} \cdot R(B_1')$$

where $B_1' = (b_2', b_3', \ldots, b_{\ell'}')$. Since $b_1 = b_1'$, $R(B) \geq R(B')$ if and only if $R(B_1) \geq R(B_1')$. By repeating the same argument for B_1 and B_1', we get that $R(B) \geq R(B')$ if and only if $R(B_2) \geq R(B_2')$, where $B_2 = (b_3, \ldots, b_\ell)$ and $B_2' = (b_3', \ldots, b_{\ell'}')$. And so on, until we get that $R(B) \geq R(B')$ if and only if $R(B_j) \geq R(B_j')$, where $B_j = (b_{j+1}, \ldots, b_\ell)$ and $B_j' = (b_{j+1}', \ldots, b_{\ell'}')$. But from Lemma 3, we know that $R(B_j) \geq R(B_j')$ if $b_{j+1} > b_{j+1}'$, and we are done. $\diamond$