Complexity of Distance Fraud Attacks in Graph-Based Distance Bounding

Rolando Trujillo-Rasua(✉)

University of Luxembourg, SnT, Luxembourg, Luxembourg
rolando.trujillo@uni.lu

Abstract. *Distance bounding* (DB) emerged as a countermeasure to the so-called *relay attack*, which affects several technologies such as RFID, NFC, Bluetooth, and Ad-hoc networks. A prominent family of DB protocols are those based on graphs, which were introduced in 2010 to resist both mafia and distance frauds. The security analysis in terms of distance fraud is performed by considering an adversary that, given a vertex labeled graph $G = (V, E)$ and a vertex $v \in V$, is able to find the most frequent n-long sequence in G starting from v (MFS problem). However, to the best of our knowledge, it is still an open question whether the distance fraud security can be computed considering the aforementioned adversarial model. Our first contribution is a proof that the MFS problem is NP-Hard even when the graph is constrained to meet the requirements of a graph-based DB protocol. Although this result does not invalidate the model, it does suggest that a *too-strong* adversary is perhaps being considered (*i.e.*, in practice, graph-based DB protocols might resist distance fraud better than the security model suggests.) Our second contribution is an algorithm addressing the distance fraud security of the tree-based approach due to Avoine and Tchamkerten. The novel algorithm improves the computational complexity $O(2^{2^n+n})$ of the naive approach to $O(2^{2n}n)$ where n is the number of rounds.

Keywords: Security · Relay attack · Distance bounding · Most frequent sequence · Graph · NP-complete · NP-hard

1 Introduction

Let us consider a little girl willing to compete with two chess grandmasters, say Fischer and Spassky. She agrees with both on playing by post and manages to use opposite-colored pieces in the games. Once the little girl receives Fisher's move she simply forwards it to Spassky and vice versa. As a result, she wins one game or draws both even though she might know nothing about chess. This problem, known as the *chess grandmaster problem*, was introduced by Conway in 1976 [7] and informally describes how relay attacks work.

In a relay attack, an adversary acts as a *passive* man-in-the-middle attacker relaying messages between the prover and the verifier during an authentication protocol. In case the adversary is *active*, the attack is known as *mafia fraud* [8]

© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2014
I. Stojmenovic et al. (Eds.): MOBIQUITOUS 2013, LNICST 131, pp. 289–302, 2014.
DOI: 10.1007/978-3-319-11569-6_23

and succeeds if the prover and the verifier complete the authentication protocol without noticing the presence of the adversary.

With the widespread deployment of contactless technologies in recent years, mafia fraud has re-emerged as a serious security threat for authentication schemes. Radio Frequency IDentification (RFID), Near Field Communication (NFC), and Passive Keyless Entry and Start Systems in Modern Cars, have been proven to be vulnerable to mafia fraud [10,13]. Other contactless technologies such as smart-cards and e-voting are also threatened by this attack [9,16].

The most promising countermeasure to thwart mafia fraud is *distance bounding* (DB) [4], that is, an authentication protocol where time-critical sessions allow to compute an upper bound of the distance between the prover and the verifier. However, this type of protocols is vulnerable to another type of fraud, the *distance fraud* [4]. Contrary to mafia fraud, distance fraud is performed by a legitimate prover, who aims to authenticate beyond the expected and allowed distance.

In 2010, graph-based DB protocols aimed at being resistant to both mafia and distance frauds were introduced [19]. This type of protocols is flexible in the sense that different graph structures can be used so as to balance memory requirements and security properties. However, neither the graph-based approach in [3] nor the one in [19] have computed their actual distance fraud security. Indeed, this analysis was left as an open problem in [19].

Contributions. In this article we address the open problem of computing the distance fraud resistance of graph-based DB protocols. We first reformulate the security model provided in [19] and define it in terms of, to the best of our knowledge, two new problems in Graph Theory. *The Most Frequent Sequence problem* (MFS problem) and its simplified version *the Binary Most Frequent Sequence problem* (Binary MFS problem).

We then provide a polynomial-time reduction of the Satisfiability problem (SAT) to the Binary MFS problem, proving that both the Binary MFS and the MFS problems are NP-Hard. This result suggests that a *too-strong* adversary is perhaps being considered by the security model, unless $P = NP$. However, the implications of our reduction goes beyond that. It also provides a clue of how to design graph-based DB protocols resistant to distance fraud.

Our next contribution is a novel algorithm to compute the distance fraud resistance of the tree-based DB protocol proposed by Avoine and Tchamkerten [3]. Our algorithm significantly reduces the time complexity of the naive approach from $O(2^{2^n+n})$ to $O(2^{2n}n)$ where n is the number of rounds. This paves the way for a fair comparison of graph-based proposals with other state-of-the-art DB protocols.

Organization. The rest of this article is organized as follows. Section 2 introduces graph-based DB protocols and the new problems Binary MFS and MFS. Related works close to the MFS problem are reviewed in Sect. 2 as well. Section 3 contains proofs on the hardness of the Binary MFS problem. The algorithm for computing the distance fraud resistance of the tree-based DB protocol is described and analyzed in Sect. 4. The discussion and conclusions are drawn in Sect. 5.

2 Preliminaries

2.1 Graph-Based Distance Bounding Protocols

Graph-based DB protocols were introduced in [19] aimed at resisting both mafia and distance frauds, yet requiring low memory to be implemented. The idea is to define a digraph $G = (V, E)$ and a starting vertex $v \in V$. Then, a challenge-response protocol (see Fig. 1) is executed where the challenges define a walk in G according to an edge labeling function $\ell_E : E \rightarrow \{0, 1\}$ and the responses are stored on the vertices according to a vertex labeling function $\ell_V : V \rightarrow \{0, 1\}$.

Algorithm 1. Graph-based distance bounding protocol

```
                 Verifier                                   Prover
                 (Secret $x$)                               (Secret $x$)
                 (Digraph $G = (V, E)$)                     (Digraph $G = (V, E)$)
                 (Starting vertex $v \in V$)                (Starting vertex $v \in V$)

                 Pick a random $N_V$                        Pick a random $N_P$
                                      $\xrightarrow{N_V}$
                                      $\xleftarrow{N_P}$
On input $PRF(K, N_P, N_V)$                                 On input $PRF(K, N_P, N_V)$
Create $\ell_V$ and $\ell_E$                                Create $\ell_V$ and $\ell_E$
$v'_0 = v$                                                  $v_0 = v$
                              for $i = 1$ to $n$
    Pick $c_i \in_R \{0, 1\}$
    Start Timer                       $\xrightarrow{c_i}$
                                                            Look for $v_i$ such that
    Stop Timer                        $\xleftarrow{\ell_V(v_i)}$   $\ell_E(v_{i-1}, v_i) = c_i$
    Look for $v'_i$ such that
    $\ell_E(v'_{i-1}, v'_i) = c_i$
Check that $\Delta t_i \leq t_{\max}$ and
$\ell_V(v'_i) = \ell_V(v_i)$
```

As shown by Fig. 1, prover and verifier exchange two nonces and use a pseudo-random function ($PRF(.)$) with a shared private key to compute two labeling functions $\ell_V : V \rightarrow \sum$ and $\ell_E : E \rightarrow \sum$ where $\sum = \{0, 1\}$. For ℓ_E it must hold that $\ell_E(u, v) \neq \ell_E(u, w)$ for every pair of different edges (u, v) and (u, w) in E. By contrast, ℓ_V is chosen randomly. After labeling the graph, n rounds of time-critical sessions are executed. At the ith round, the verifier sends the binary challenge c_i. Then, the prover answers with $\ell_V(v_i)$ where v_i holds that $(v_{i-1}, v_i) \in E$ and $\ell_E(v_{i-1}, v_i) = c_i$. Note that v_0 is the starting vertex v. At the end of the n time-critical sessions, the verifier checks all prover's responses ($\ell_V(v_i)$) and the round-trip-times (Δt_i), which should be below some threshold $t_{\max}$. Intuitively, the lower $t_{\max}$ the closer the prover to the verifier is expected to be.

Two graph-based DB protocols exist; the tree-based approach [3] and the Poulidor protocol [19]. As suggested by its name, the former uses a tree of depth

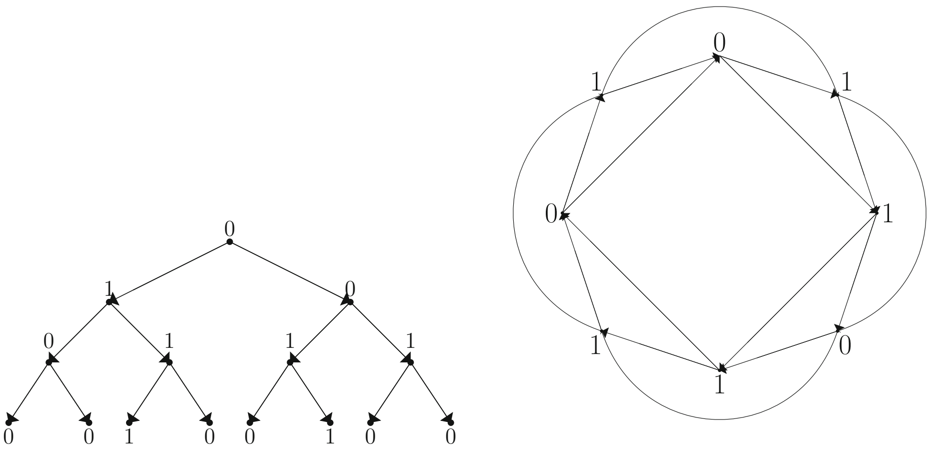
(a) Tree-based's graph for 3 rounds
(b) Poulidor's graph for 4 rounds

Fig. 1. Graph structures used by the tree-based and Poulidor approaches. Vertices of both graphs have been randomly labeled.

n and $2^{n+1}-1$ nodes (see Fig. 1(a)). By contrast, Poulidor uses a graph structure with $2n$ nodes only in order to reduce memory requirements (see Fig. 1(b)). Both have proven to resist mafia fraud better than other DB protocols such as [12,20]. Its resistance to distance fraud, however, is still an open problem.

There exist other DB protocols, more computationally demanding, based on signatures and/or a final extra slow phase [4,21]. Others simply could be plugged into most DB protocols such as [22,23]. The interested reader could refer to [2] for more details.

2.2 Distance Fraud Security Analysis

The security analysis in terms of distance fraud is usually performed within a well-known framework proposed by Avoine et al. [2]. In this framework, a distance fraud adversary uses the *early-reply strategy* to defeat the DB protocol. This strategy consists on sending the bits answer in advance (*i.e.,* before receiving the challenges.) Doing so, the adversary simulates to be closer than really is, and its success probability is lower-bounded by $1/2^n$.

In [19], the best early-reply strategy against what they called a *family* of DB protocols is defined. This *family* includes graph-based DB protocols. However, their definition is too generic to be used for simply analyzing graph-based DB protocols. Therefore, we reformulate it here in terms of a new problem in Graph Theory. The problem is named Binary MFS problem (see Definition 2) and is based on its more general version MFS problem (see Definition 1).

Definition 1 (The most frequent sequence problem (MFS problem)). *Let $G = (V, E)$ be a vertex-labeled digraph where Σ and $\ell : V \to \Sigma$ are the set* *of vertex labels and the labeling function respectively. For a label sequence $t = t_1 t_2 ... t_k$, $occ_v^G(t)$ denotes the number of walks $v_1 v_2 ... v_k$ in G such that $v_1 = v$ and $\forall i \in \{1, ..., k\}$ $(\ell(v_i) = t_i)$. The MFS problem consists on finding, given the triple (G, v, k), the* most frequent sequence *of size k defined as* $\arg\max_{t \in \Sigma^k}(occ_v^G(t))$.

Definition 2 (Binary MFS problem). *The Binary MFS problem is an MFS problem where G is constrained to use a binary vertex set ($\sum = \{0, 1\}$) and the out-degree of every vertex should be at most* 2.

Example. Either the graph in Fig. 1(a) or the one in Fig. 1(b) can be the input of the Binary MFS problem. Assuming $k = 4$ and the starting vertex as the top one, the most frequent sequences for the tree in Fig. 1(a) is 0010 (occurs 3 times), and for the graph in Fig. 1(b) is 0101 (occurs 4 times).

To successfully apply a distance fraud attack against a graph-based DB protocol with n time-critical sessions, the best adversary's strategy consists of: (i) solving the Binary MFS problem defined by the triple $(G, v, n+1)$ and finding the most frequent sequence $t_0 t_1 ... t_n$, (ii) sending $t_1 ... t_n$ in advance to the verifier as the responses to the n verifier's challenges. By this strategy, the adversary's success probability is maximized to $\frac{occ_v^G(t)}{2^n}$ [19]. Coming back to the previous example, the adversary success probability of the tree-based approach defined by Fig. 1(a) is 3/8, which is higher than the expected lower bound 1/8.

Definition 3 (Distance fraud success probability). *Let $\prod$ be a graph-based DB protocol with n time-critical sessions that uses the vertex-labeled digraph $G = (V, E)$ and $v \in V$ as the starting vertex. Let $M_{G,v,n}$ be a random variable on the sample space of all labeling functions $\ell : V \to \Sigma$ that outputs the maximum value* $\max(occ_v^G(t))$ *where $t \in \{0, 1\}^{n+1}$. The* distance fraud success probability *of an adversary against $\prod$ is defined as $\frac{E(M_{G,v,n})}{2^n}$ where $E(M_{G,v,n})$ represents the expectation of the random variable $M_{G,v,n}$.*

Note that, following the design of graph-based DB protocols, Definition 3 considers that G is randomly labeled at each execution of the protocol.

To the best of our knowledge, computing distance fraud security according to Definition 3 has been only addressed in its seminal work [19]. Apparently, the problem is one of those problems that remain intractable even if $P = NP$ because all, or almost all, the labeling functions should be considered. For this reason, an upper bound was proposed in [19] and the exact distance fraud security was left as an open problem. We have shown that this problem depends on a problem named the MFS problem and, in particular, on the Binary MFS problem. Below, we review some work related to them.

2.3 Review on Frequent Sequences Problems

Sequential Pattern Mining is a well-studied field introduced by Agrawal and Srikant [1] in 1995. Given a databases of transactions (*e.g.*, customer transactions, medical records, web sessions, etc.) the problem consists on discovering all the sequential patterns with some minimum support. The support of a pattern is defined as the number of data-sequence within the database that are contained in the pattern.

The sequential pattern mining problem is $\#P$-complete [24] and several variants of it exist. For instance, Mannila et al. say that two events are connected if they are close enough in terms of time [15]. They define an *episode* as a collection of connected events and the problem is to find frequently occurring episodes in a sequence. A simpler variant, known as the *most common subsequent problem*, was introduced by Campagna and Pagh [5]. The most common subsequent problem does not consider time-stamped events. Instead, it aims to find all the label sequences in a vertex-labeled acyclic graph that appear more often. Other variants have arisen from complex applications namely, telecommunication, market analysis, and DNA research. We refer the reader to [6] for an extensive survey on this subject.

Frequent paths on a graph have also been used to define Kernel functions [17, 18]. Kernel functions has applicability in chemoinformatics and bioinformatics where objects are mapped to a feature space. In this case, the feature space representation is the number of occurrences of vertex-labeled paths and the problem is to infer the graph from such a feature vector. This problem has been proven to be NP-Hard even for trees of bounded degree [18].

It can be seen that the MFS problem is different to the sequential pattern mining problems and its nature is obviously different to the one of Kernel methods. On one hand, sequential pattern mining is an enumeration problem while MFS is just a search problem. On the other hand, the MFS problem requires all walks to begin from a given vertex and the size of the sequences should be equal. As in [5], the time dimension is not considered.

3 On the Hardness of the Binary MFS Problem

Binary MFS is a search problem that looks for the most frequent sequence of length k in a vertex-labeled digraph G starting from a given vertex v (see Definition 2). Intuitively, all or almost all the walks in G starting from v should be analyzed in order to find such a sequence, which means that Binary MFS might not be in the complexity class P. However, we cannot even state that Binary MFS is in $NP - P$ since it is not trivial how to *check* a solution in polynomial time. Nevertheless, we prove in this Section (see Theorem 1) that the general Boolean Satisfiability problem (SAT) reduces to Binary MFS. Therefore, Binary MFS can be considered NP-Hard even thought it may not even be in NP [11].

Definition 4 (SAT). *Let $x = (x_1, x_2, ..., x_n)$ be a set of boolean variables and $c_1(x), c_2(x), ..., c_m(x)$ be a set of clauses where $c_i(x)$ is a disjunction of literals. The Boolean satisfiability problem (SAT) consists on deciding whether there exists an assignment for the boolean variables x such that the function $f_{SAT} = c_1(x) \wedge c_2(x) \wedge ... \wedge c_m(x) = 1$.*

Algorithm 2 shows our reduction from SAT to an instance of the Binary MFS problem. First, it creates a binary tree T of depth $\lceil \log m \rceil$ with m leafs

c_1, c_2, ..., c_m[1], and a graph $G' = (V', E')$ where $V' = \{u_0^2, v_0^2, ..., u_0^n, v_0^n\}$ and $E' = \{(x,y) | \exists k \in \{2, ..., n-1\}(x \in \{u_0^k, v_0^k\} \wedge y \in \{u_0^{k+1}, v_0^{k+1}\})\}$. The graph $G = (V, E)$ is initialized with the two connected components T and G'. In addition, V is increased with the vertices u_i^j and v_i^j where $i \in \{1, ..., m\}$ and $j \in \{1, ..., n\}$. Then, for each clause c_i and each variable x_j $(j < n)$, the vertex u_i^j is connected with u_0^{j+1} and v_0^{j+1} if $x_j \in c_i(x)$, with u_i^{j+1} and v_i^{j+1} otherwise. Similarly, the vertex v_i^j is connected with u_0^{j+1} and v_0^{j+1} if $\neg x_j \in c_i(x)$, with u_i^{j+1} and v_i^{j+1} otherwise. Finally, for every $i \in \{1, ..., m\}$: (i) the vertices u_i^n and v_i^n are removed together with their incident edges, (ii) the edges (u_i^{n-1}, u_0^n) and (v_i^{n-1}, u_0^n) are added if $x_n \in c_i(x)$, (iii) if $\neg x_n \in c_i(x)$ the added edges are (u_i^{n-1}, v_0^n) and (v_i^{n-1}, v_0^n). The vertex-label function is simply defined as a function that outputs 0 on input v_i^j for every $i \in \{0, 1, ..., m\}$ and $j \in \{1, ..., n\}$, outputs 1 otherwise.

To better illustrate Algorithm 2, Fig. 2 shows an example of its output for a given SAT instance. Note that, Algorithm 2 does not consider tautologies such as the empty clause or one containing $x \vee \neg x$.

Algorithm 2. Reduction from the SAT problem

```
Require: A SAT instance where $x = (x_1, x_2, ..., x_n)$ are the boolean variables and
    $c_1(x), c_2(x), ..., c_m(x)$ are the set of clauses
 1: Let $G = (V, E)$ be a digraph with just one vertex named root
 2: From the root, a directed binary tree with $m$ leafs is created such that all the leafs
        are at the same depth. The leaf vertices are denoted as $c_1, c_2, ..., c_m$
 3: Let $G' = (V', E')$ be a digraph where $V' = \{u_0^2, v_0^2, ..., u_0^n, v_0^n\}$ and $E' =$
        $\{(x,y) | \exists k \in \{2, ..., n-1\}(x \in \{u_0^k, v_0^k\} \wedge y \in \{u_0^{k+1}, v_0^{k+1}\})\}$
 4: Set $G = G \cup G'$
 5: for all vertex $c_i$ do
 6:     Set $V = V \cup \{u_i^1, v_i^1, u_i^2, v_i^2, ..., u_i^n, v_i^n\}$
 7:     Set $E = E \cup \{(c_i, u_i^1), (c_i, v_i^1)\}$
 8:     for all $j \in \{1, 2, ..., n-1\}$ do
 9:         if $x_j \in c_i(x)$ then
10:             Set $E = E \cup \{(u_i^j, u_0^{j+1}), (u_i^j, v_0^{j+1})\}$ and $E = E \cup \{(v_i^j, u_i^{j+1}), (v_i^j, v_i^{j+1})\}$
11:         else if $\neg x_j \in c_i(x)$ then
12:             Set $E = E \cup \{(u_i^j, u_i^{j+1}), (u_i^j, v_i^{j+1})\}$ and $E = E \cup \{(v_i^j, u_0^{j+1}), (v_i^j, v_0^{j+1})\}$
13:         else
14:             Set $E = E \cup \{(u_i^j, u_i^{j+1}), (u_i^j, v_i^{j+1})\}$ and $E = E \cup \{(v_i^j, u_i^{j+1}), (v_i^j, v_i^{j+1})\}$
15:     Remove $u_i^n$ and $v_i^n$ from $G$
16:     if $x_n \in c_i(x)$ then Set $E = E \cup \{(u_i^{n-1}, u_0^n), (v_i^{n-1}, u_0^n)\}$
17:     if $\neg x_n \in c_i(x)$ then Set $E = E \cup \{(u_i^{n-1}, v_0^n), (v_i^{n-1}, v_0^n)\}$
18: Create vertex-label function $\ell_V(.)$ such that $\forall i \in \{0, 1, ..., m\}, j \in$
        $\{1, ..., n\}(\ell_V(v_i^j) = 0)$, $\ell_V(.)$ outputs 1 otherwise.
19: Return $G$ and $\ell_V$ as result.
```

[1] The leafs are intentionally labeled by using the same clause names.

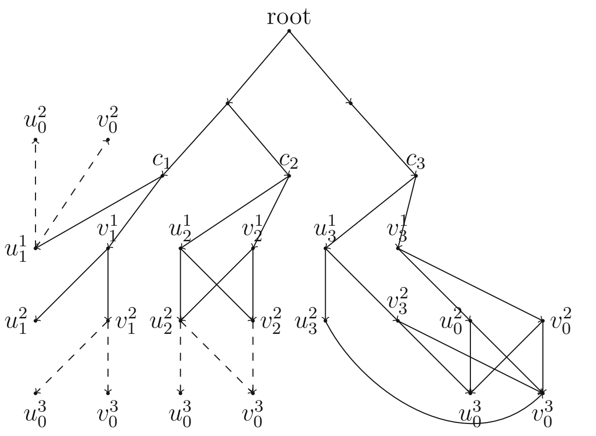

Fig. 2. The resulting graph when applying Algorithm 2 on input the boolean formula $(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$. For the sake of a good visualization some nodes have been "cloned", however, they actually represent a single node in the graph. Such "cloned" nodes can be easily identified as the ones with dashed incident edges.

Lemma 1. *The longest walk in G starting from the* root *vertex has length $n + \lceil \log m \rceil$ and ends either at u_0^n or v_0^n.*

Proof. Let $w = w_0 ... w_k$ be a walk in $G = (V, E)$ starting from the *root* vertex. Let us assume that w is maximal in the sense that w_k does not have out-going edges. According to Algorithm 2, w_k does not have out-going edges only if $w_k \in \{u_0^n, v_0^n\}$ or $w_k \in \{u_1^{n-1}, v_1^{n-1}, u_2^{n-1}, v_2^{n-1}, ..., u_m^{n-1}, v_m^{n-1}\}$ (see Step 15 of Algorithm 2). Therefore, either the longest walk ends at u_0^n or v_0^n and its length is $n + \lceil \log m \rceil$ or its length is $n - 1 + \lceil \log m \rceil$. The proof concludes by remarking that there must exist at least one walk ending at u_0^n or v_0^n unless all the clauses are empty, which is a tautology not-considered in SAT. □

Lemma 2. *Let $s = s_0 s_1 ... s_{n+\lceil \log m \rceil}$ and $t = t_0 t_1 ... t_{n+\lceil \log m \rceil}$ two different maximal length walks in G that start from the* root *vertex. Then, $\forall k \in \{0, ..., n + \lceil \log m \rceil\}(\ell_V(s_k) = \ell_V(t_k)) \Rightarrow \exists i \neq j(c_i \in s \wedge c_j \in t)$.*

Proof. According to Algorithm 2, there exist $i, j \in \{1, ..., m\}$ such that $c_i = s_{\lceil \log m \rceil}$ and $c_j = t_{\lceil \log m \rceil}$. In addition, every vertex s_k (resp. t_k) where $\lceil \log m \rceil < k < n + \lceil \log m \rceil$ is either $u_i^{k-\lceil \log m \rceil}$ (resp. $u_j^{k-\lceil \log m \rceil}$) or $v_i^{k-\lceil \log m \rceil}$ (resp. $v_j^{k-\lceil \log m \rceil}$).

Now, according to the vertex-label function ℓ_V, if $\forall k \in \{0, ..., n-1+\lceil \log m \rceil\}$ $(\ell_V(s_k) = \ell_V(t_k))$, then $\forall k \in \{\lceil \log m \rceil + 1, ..., n-1+\lceil \log m \rceil\}(s_k = v_i^{k-\lceil \log m \rceil} \Leftrightarrow t_k = v_j^{k-\lceil \log m \rceil})$. Similarly, if $\ell_V(s_{n+\lceil \log m \rceil}) = \ell_V(t_{n+\lceil \log m \rceil})$ then $s_{n+\lceil \log m \rceil} = v_0^n \Leftrightarrow t_{n+\lceil \log m \rceil} = v_0^n$ (see Lemma 1). Therefore, $i = j \Rightarrow s = t$, which is a contradiction. □

Theorem 1. *The Binary MFS problem is NP-Hard.*

Proof. Let $\prod$ be an instance of the SAT problem and let G be the graph obtained by applying Algorithm 2 on input $\prod$. Let $\prod'$ be the problem of finding the most frequent sequence of length $n + 1 + \lceil \log m \rceil$ in G starting from the vertex root. Given a solution s for $\prod'$, our aim is to prove that a true assignment for $\prod$ exists (and can be found) in polynomial time if and only if s appears m times in G. Doing so, $\prod$ is proven to be polynomially reducible to $\prod'$, which is a Binary MFS problem.

First, let us assume that the most frequent sequence $s = s_0 s_1 ... s_{n+\lceil \log m \rceil}$ in G occurs exactly m times. Let $w_0 w_1 ... w_{n+\lceil \log m \rceil}$ be a walk such that $\forall i \in \{0, ..., n + \lceil \log m \rceil\}(\ell_V(w_i) = s_i)$. By Algorithm 2, there must exist $i \in \{1, ..., m\}$ such that $w_{\lceil \log m \rceil} = c_i$. Let $w_{k+\lceil \log m \rceil}$ be the vertex such that $w_{k+\lceil \log m \rceil} \notin \{u_0^k, v_0^k\}$ and $w_{k+1+\lceil \log m \rceil} \in \{u_0^{k+1}, v_0^{k+1}\}$. Note that, such a vertex exists due to Lemma 1 and Algorithm 2. According to such an algorithm, either $w_{k+\lceil \log m \rceil} = u_i^k$ and $c_i(x)$ contains the literal x_k or $w_{k+\lceil \log m \rceil} = v_i^k$ and $c_i(x)$ contains the literal $\neg x_k$. Therefore, if $w_{k+\lceil \log m \rceil} = u_i^k$ then $x_k = s_{k+\lceil \log m \rceil} = \ell_V(u_i^k) = 1$ satisfies the clause $c_i(x)$, otherwise $x_k = s_{k+\lceil \log m \rceil} = \ell_V(v_i^k) = 0$ does. Consequently, the assignment $x_j = s_{j+\lceil \log m \rceil}$ $\forall j \in \{1, ..., n\}$ satisfies $c_i(x)$.

Considering that s appears m times, then by Lemma 2 we can conclude that all the clauses are satisfied by such assignment, whereupon we finish the first part of this proof.

Now, let $x = (y_1, ..., y_n)$ be a true assignment for $\prod$. Let us consider the induced sub-graph G_i formed by the vertex c_i and all the other vertices reachable from c_i. By design of Algorithm 2, there exists a walk $w = w_0 w_1 ... w_{n-1}$ in G_i such that $\forall k \in \{1, ..., n-1\}(y_k = \ell_V(w_k))$ and $w_0 = c_i$. In addition, if y_j satisfies clause $c_i(x)$ (*i.e.*, $y_j = 1 \wedge x_j \in c_i(x)$ or $y_j = 0 \wedge \neg x_j \in c_i(x)$), then according to Algorithm 2 either $(x_j \in c_i(x) \wedge \{(u_i^j, u_0^{j+1}), (u_i^j, v_0^{j+1})\} \in E)$ or $(\neg x_j \in c_i(x) \wedge \{(v_i^j, u_0^{j+1}), (v_i^j, v_0^{j+1})\} \in E)$. Consequently, it must hold that $w_{n-1} \in \{u_0^{n-1}, v_0^{n-1}\}$ if and only if $(x_1, ..., x_{n-1}) = (y_1, ..., y_{n-1})$ satisfies $c_i(x)$. If $(y_1, ..., y_{n-1})$ does satisfies $c_i(x)$, the walk $w = w_0 w_1 ... w_{n-1} w_n$ where $w_n = u_0^n$ if $y_n = 1$ and $w_n = v_0^n$ if $y_n = 0$ holds that $\forall k \in \{1, ..., n\}(y_k = \ell_V(w_k))$. On the other hand, if $(y_1, ..., y_{n-1})$ does not satisfies $c_i(x)$, then $y_n = 1 \Rightarrow x_n \in c_i(x)$ and $y_n = 0 \Rightarrow \neg x_n \in c_i(x)$. According to Steps 18 and 20 of Algorithm 2, $w = w_0 w_1 ... w_{n-1} w_n$ where $w_n = u_0^n$ if $y_n = 1$ and $w_n = v_0^n$ if $y_n = 0$ is a walk holding that $\forall k \in \{1, ..., n\}(y_k = \ell_V(w_k))$. As a conclusion, for every $i \in \{1, ..., m\}$ there exists a walk passing through c_i that generates the sequence $\underbrace{11...11}_{\lceil \log m \rceil + 1} y_1 ... y_n$. This result together with Lemma 2 conclude that such a sequence repeats exactly m times. □

Corollary 1. *The MFS problem is NP-Hard.*

4 Distance Fraud Analysis for the Tree-Based Approach

In this section, the problem of computing the distance fraud resistance of the tree-based DB protocol [3] is addressed. A naive algorithm to solve this problem consists on analyzing all the labeling functions for a full binary tree of depth n and then computing the most frequent sequence for each labeling function (see Definition 3). This results in a time-complexity of $O(2^{2^n+n})$, which is unfeasible even for small values of n. We propose an algorithm that reduces this time-complexity to $O(2^{2n}n)$. Although still exponential, it might be used up to reasonable values of n (*e.g.*, $n = 32$).

For the sake of clarity, we first adapt Definition 3 to the context of the tree-based proposal.

Problem 1. [Tree-based distance fraud problem] Let $\prod$ be a tree-based DB protocol with n time-critical sessions that uses the full binary tree $T = (V, E)$ and $root \in V$ as the starting vertex. Let M_n be a random variable on the sample space of all labeling functions $\ell : V \to \{0, 1\}$ that outputs the maximum value $\max(occ^T_{root}(t))$ where $t \in \{0, 1\}^{n+1}$. The *tree-based distance fraud problem* consists on finding the expectation of the random variable M_n.

Theorem 2. *Let m and n be two positive integers. Let T^m_n be a tree such that: (i) the root has $2m$ children and (ii) each root's children is the root of a full binary tree of depth $n-1$. Let M^m_n be the random variable on the sample space of all binary vertex-labeling functions over T^m_n that outputs the maximum value $\max(occ^{T^m_n}_{root}(t))$. The expectation of the random variable M_n can be computed as follows:*

$$E(M_n) = E(M^1_n) = \sum_{i=1}^{2^n} \left(\Pr(M^1_n < i+1) - \Pr(M^1_n < i)\right) i.$$

where

$$\Pr(M^m_n < x) = \sum_{i=0}^{2m} \frac{\binom{2m}{i}}{2^{2m}} \left(\Pr(M^i_{n-1} < x)\Pr(M^{2m-i}_{n-1} < x)\right).$$

and

$$\Pr(M^m_n < x) = \begin{cases} 0 & \textit{if} \quad x = 1 \\ 0 & \textit{if} \quad n = 1 \wedge m > x \\ \frac{1}{2^{2m}}\left(\binom{2m}{m} + 2\sum\limits_{i=m+1}^{x-1}\binom{2m}{i}\right) & \textit{if} \quad n = 1 \wedge m \leq x \leq 2m \\ 1 & \textit{if} \quad n = 1 \wedge 2m < x \\ 1 & \textit{if} \quad m = 0 \end{cases}$$

Proof. A full binary tree of depth n can be denoted as T^1_n and the random variable M_n is equivalent to M^1_n. Therefore, in what follows, we focus on computing the expectation of the random variable M^m_n.

Let us consider now a labeling function ℓ over T_n^m. Let V_0 and V_1 be the set of children of the T_n^m's root labeled with 0 and 1 respectively. Let $C_1^0, C_2^0, ..., C_{2|V_0|}^0$ and $C_1^1, C_2^1, ..., C_{2|V_1|}^1$ be the subtrees rooted in the children of the vertices in V_0 and V_1 respectively. Let X be a labeled tree whose root is labeled with 0 and the root's children are the full binary trees $C_1^0, C_2^0, ..., C_{2|V_0|}^0$. In the same vein, Y is defined as a labeled tree whose root is labeled with 1 and the root's children are the full binary trees $C_1^1, C_2^1, ..., C_{2|V_1|}^1$. It can be noted that, if a sequence $t = t_1...t_n$ occurs exactly k times either in X or Y, then the sequence $t = t_0t_1...t_n$ where t_0 is the label of T_n^m's root also appears exactly k times in T_n^m. Therefore, taking into account that $X = T_{n-1}^{|V_0|}$ and $Y = T_{n-1}^{|V_1|}$, the following recurrent result can be obtained:

$$\begin{aligned}\Pr(M_n^m < x) &= \sum_{i=0}^{2m} \Pr(|V_0| = i)\left(\Pr(M_{n-1}^i < x)\Pr(M_{n-1}^{2m-i} < x)\right)\\ &= \sum_{i=0}^{2m} \frac{\binom{2m}{i}}{2^{2m}}\left(\Pr(M_{n-1}^i < x)\Pr(M_{n-1}^{2m-i} < x)\right). \end{aligned} \tag{1}$$

Equation 1 shows that $\Pr(M_n^m < x)$ could be computed recursively. To do so, stop conditions must be found as follows:

$$\Pr(M_n^m < x) = \begin{cases} 0 & \text{if } \quad x = 1 \\ 0 & \text{if } \quad n = 1 \wedge m > x \\ \frac{1}{2^{2m}}\left(\binom{2m}{m} + 2\sum\limits_{i=m+1}^{x-1}\binom{2m}{i}\right) & \text{if } \quad n = 1 \wedge m \leq x \leq 2m \\ 1 & \text{if } \quad n = 1 \wedge 2m < x \\ 1 & \text{if } \quad m = 0 \end{cases} \tag{2}$$

Let us analyze $\Pr(M_1^m < x)$, which is the less trivial stop condition in Eq. 2. Since T_1^m has depth 1 and $2m$ children, T_1^m generates $2m$ sequences, p of them ending with 0 and q with 1. Consequently, $M_1^m = \max(p, q) \geq m$ and thus, $\Pr(M_1^m < x) = 0$ if $x < m$. Similarly $M_1^m \leq 2m$, which implies that $\Pr(M_1^m < x) = 1$ if $x > 2m$. Finally, let us assume that $m \geq x \geq 2m$. In this case, $M_1^m < x$ holds if $M_1^m \in \{m, m+1, ..., x-1\}$, therefore, $\Pr(M_1^m < x) = \sum\limits_{i=m}^{x-1} \Pr(M_1^m = i)$ where $\Pr(M_1^m = i) = \frac{\binom{2m}{m}}{2^{2m}}$ if $i = m$, otherwise $\Pr(M_1^m = i) = 2\frac{\binom{2m}{i}}{2^{2m}}$. This yields to $\Pr(M_1^m < x) = \frac{1}{2^{2m}}\left(\binom{2m}{m} + 2\sum\limits_{i=m+1}^{x-1}\binom{2m}{i}\right)$ if $m \geq x \geq 2m$.

The proof concludes by using the definition of expectation for a discrete variable together with Eqs. 1 and 2. □

Time-complexity analysis. The result provided by Theorem 2 can be implemented by a dynamic algorithm, meaning that a three-dimensional matrix will dynamically store the values of $\Pr(M_n^m < x)$ and will use them when needed.

In the worst case, the algorithm requires to fill the whole matrix, which results in a time-complexity of $O(2^n \times n \times 2^n) = O(2^{2n}n)$.

5 Discussion and Conclusions

Before the introduction of graph-based DB protocols, computing resistance to distance fraud was not a big issue (*e.g.*, Hancke and Kuhn [12] and Kim and Avoine [14] proposals.) Actually, the well-known early-reply strategy used to analyze distance fraud security implicitly assumes that the adversary is able to compute the *best* answer without knowing the challenges and within a "reasonable" time frame. In this article, however, we have shown that this assumption might not hold for graph-based DB protocols by proving that the Binary MFS problem is NP-Hard. This opens two interesting research questions: (i) What instances of the Binary MFS problem are actually hard to solve? (ii) In a practical setting, does the adversary have enough time to solve a probably exponential problem between the end of the slow phase and the beginning of the fast phase? (iii) What kinds of heuristics can be used and what would the implications be?

Even though we do not give answers to those questions, we provide a clue of how to build graph-based DB protocols resistant to distance fraud. As indicate our reduction from the SAT problem, a good strategy is to label the vertices of G as follows: if the vertices u and v have incident edges from the same vertex, then $\forall b \in \{0,1\}(\ell_V(u) = b \Leftrightarrow \ell_V(v) = b \oplus 1)$. Doing so, G is likely to generate all the sequences $\{0,1\}^n$ just once, in which case its resistance to distance fraud achieves the lower bound $1/2^n$. As a consequence, we conjecture that the best graph DB protocol in terms of mafia fraud constrained to have no more than certain number of nodes, is also the best in terms of distance fraud. Note that, this conjecture becomes trivial if no limit on the size of the graph is considered.

This article has also addressed the problem of computing the distance fraud security of the tree-based DB proposal [3]. This is an inherent exponential problem since a graph with N nodes can be labeled in 2^N different ways. The tree-based proposal uses a tree with 2^n nodes and thus 2^{2^n} labelling functions exists. However, we provide an algorithm that avoids considering all the labelling functions and has a time complexity of $O(2^{2n}n)$, which is significantly better than the naive approach with $O(2^{2^n+n})$. This result makes realistic the challenge of computing the distance fraud security of the two graph-based DB protocols proposed up-to-date; the tree-based approach [3] by using the proposed algorithm ($O(2^{2n}n)$), and the Poulidor protocol [19] by simply using a brute-force algorithm ($O(2^{3n})$). Doing so, both can be fairly compared with other state-of-the-art DB protocols. Such a challenge is out of the scope of this article and is left as future work, though.

Acknowledgments. The author thanks to Gildas Avoine, Sjouke Mauw, Juan Alberto Rodriguez-Velazquez, and Alejandro Estrada-Moreno for their invaluable comments and feedback.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: ICDE, pp. 3–14 (1995)
2. Avoine, G., Bingöl, M.A., Kardaş, S., Lauradoux, C., Martin, B.: A framework for analyzing RFID distance bounding protocols. J. Comput. Secur. **19**(2), 289–317 (2011)
3. Avoine, G., Tchamkerten, A.: An efficient distance bounding RFID authentication protocol: balancing false-acceptance rate and memory requirement. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 250–261. Springer, Heidelberg (2009)
4. Brands, S., Chaum, D.: Distance bounding protocols. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 344–359. Springer, Heidelberg (1994)
5. Campagna, A., Pagh, R.: On finding frequent patterns in event sequences. In: ICDM '10, pp. 755–760 (2010)
6. Chand, C., Thakkar, A., Amit, G.: Sequential pattern mining: Survey and current research challenges. Int. J. Soft Comput. Eng. 2(1) (2012)
7. Conway, J.H.: On Numbers and Games, 2nd edn. AK Peters Ltd., Natick (2000)
8. Desmedt, Y.G., Goutier, C., Bengio, S.: Special uses and abuses of the Fiat Shamir passport protocol. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 21–39. Springer, Heidelberg (1988)
9. Drimer, S., Murdoch, S.J.: Keep your enemies close: distance bounding against smartcard relay attacks. In: USINEX, pp. 1–16 (2007)
10. Francis, L., Hancke, G., Mayes, K., Markantonakis, K.: Practical NFC peer-to-peer relay attack using mobile phones. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 35–49. Springer, Heidelberg (2010)
11. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, NY (1979)
12. Hancke, G.P., Kuhn, M.G.: An RFID distance bounding protocol. In: SECURECOMM, pp. 67–73 (2005)
13. Hancke, G.P., Kuhn, M.G.: Attacks on time-of-flight distance bounding channels. In: WiSec '08, pp. 194–202 (2008)
14. Kim, C.H., Avoine, G.: RFID distance bounding protocols with mixed challenges. IEEE Trans. Wireless Commun. **10**(5), 1618–1626 (2011)
15. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data Min. Knowl. Discov. **1**(3), 259–289 (1997)
16. Oren, Y., Wool, A.: Relay attacks on RFID-based electronic voting systems. Cryptology ePrint Archive, Report 2009/422 (2009)
17. Shimizu, M., Nagamochi, H., Akutsu, T.: Enumerating tree-like chemical graphs with given upper and lower bounds on path frequencies. BMC Bioinform. **12**, 1–9 (2011)
18. Akutsu, T., Tatsuya, D., Fukagawa, D., Jansson, J., Sadakane, K.: Inferring a graph from path frequency. Discrete Appl. Math. **160**(10–11), 1416–1428 (2012)
19. Trujillo-Rasua, R., Martin, B., Avoine, G.: The poulidor distance-bounding protocol. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 239–257. Springer, Heidelberg (2010)
20. Kim, C.H., Avoine, G.: RFID distance bounding protocol with mixed challenges to prevent relay attacks. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 119–133. Springer, Heidelberg (2009)
21. Singelée, D., Preneel, B.: Distance bounding in noisy environments. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572, pp. 101–115. Springer, Heidelberg (2007)

22. Xin, W., Yang, T., Tang, C., Hu, J., Chen, Z.: A distance bounding protocol using error state and punishment. In: IMCCC, pp. 436–440 (2011)
23. Munilla, J., Peinado, A.: Distance bounding protocols for RFID enhanced by using void-challenges and analysis in noisy channels. Wirel. Commun. Mob. Comput. **8**(9), 1227–1232 (2008)
24. Yang, G.: The complexity of mining maximal frequent itemsets and maximal frequent patterns. In: KDD '04, pp. 344–353 (2004)