Improved Protocols and Hardness Results for the Two-Player Cryptogenography Problem*

Benjamin Doerr[1] and Marvin Künnemann[2]

1 **LIX, École Polytechnique, Palaiseau, France**
`doerr@lix.polytechnique.fr`

2 **Max Planck Institute for Informatics and Saarbrücken Graduate School of Computer Science, Saarbrücken, Germany**
`marvin@mpi-inf.mpg.de`

Abstract

The cryptogenography problem, introduced by Brody, Jakobsen, Scheder, and Winkler (ITCS 2014), is to collaboratively leak a piece of information known to only one member of a group (i) without revealing who was the origin of this information and (ii) without any private communication, neither during the process nor before. Despite several deep structural results, even the smallest case of leaking one bit of information present at one of two players is not well understood. Brody et al. gave a 2-round protocol enabling the two players to succeed with probability $1/3$ and showed the hardness result that no protocol can give a success probability of more than $3/8$.

In this work, we show that neither bound is tight. Our new hardness result, obtained by a different application of the concavity method used also in the previous work, states that a success probability of better than 0.3672 is not possible. Using both theoretical and numerical approaches, we improve the lower bound to 0.3384, that is, give a protocol leading to this success probability. To ease the design of new protocols, we prove an equivalent formulation of the cryptogenography problem as solitaire vector splitting game. Via an automated game tree search, we find good strategies for this game. We then translate the splits that occurred in this strategy into inequalities relating position values and use an LP solver to find an optimal solution for these inequalities. This gives slightly better game values, but more importantly, also a more compact representation of the protocol and a way to easily verify the claimed quality of the protocol.

Unfortunately, already the smallest protocol we found that beats the previous $1/3$ success probability takes up 16 rounds of communication. The protocol leading to the bound of 0.3384 even in a compact representation consists of 18248 game states. These numbers suggest that the task of finding good protocols for the cryptogenography problem as well as understanding their structure is harder than what the simple problem formulation suggests.

1998 ACM Subject Classification G.2.m Miscellaneous

Keywords and phrases randomized protocols, anonymous communication, computer-aided proofs, solitaire games

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.150

1 Introduction

Motivated by a number of recent influential cases of whistle-blowing, Brody, Jakobsen, Scheder, and Winkler [2] proposed the following *cryptogenography problem* as model for anonymous information disclosure in public networks. We have k players (potential information leakers).

* An extended online version of this article can be accessed at [4].

A random one of them holds a secret, namely a random bit. All other players only know that they are not the secret holder. Now without any non-public communication, the players aim at both making the secret public and hiding the identity of the secret holder. More precisely, we are looking for a (fully public) communication protocol in which the players as only form of communication broadcast bits, which may depend on public information (including all previous communication), private knowledge (with respect to the secret), and a private source of randomness. After this phase of communication, the protocol outputs a single bit depending solely on all data sent in the communication phase. The complete protocol (regulating the communication and the output function) and all communication is public, and is monitored by an eavesdropper who aims at identifying the secret owner. We say that a run of the protocol is a success for the players, if the protocol output is the secret bit and the eavesdropper fails to identify the secret owner; otherwise it is a success for the eavesdropper. Since everything is public, optimal strategies for the eavesdropper are easy to find (see below). We shall therefore always assume that the eavesdropper plays an optimal strategy. The players' success probability (for a given protocol) then is the probability (taken over the random decisions of the players and the random initial secret distribution) that simultaneously (i) the protocol outputs the true secret and (ii) an optimal eavesdropper does not blame the secret holder.

It is immediately clear that some positive (players') success probability is easy to obtain. A protocol without any communication and outputting a random bit achieves a success probability of $\frac{1}{2} - \frac{1}{2k}$ (the eavesdropper has no strictly better alternative than guessing a random player). Surprisingly, Brody et al. could show that the players, despite the complete absence of private communication, can do better. For two players, they present a protocol having a success probability of $\frac{1}{3}$ (instead of the trivial $\frac{1}{4}$). For k sufficiently large, they present a protocol with success probability 0.5644. They also show two hardness results, namely that a success probability of more than $\frac{3}{4}$ cannot be obtained, regardless of the number of players, and that $\frac{3}{8}$ is an upper bound for the two-player case. While all these results are easy to state, they build on deep analyses of the cryptogenography problem, in particular, on clever reformulations of the problem in terms of certain convex combinations of secret distributions (protocol design) and functions that are concave on a certain infinite set of two-dimensional subspaces ("allowed planes") of the set of secret distributions (hardness results).

The starting point for our work is the incomplete understanding of the two-player case. While the gap between upper and lower bound of $\frac{3}{8} - \frac{1}{3} \approx 0.04167$ is small, our impression is that the current-best protocol achieving the $\frac{1}{3}$ success probability in two rounds together with the abstract hardness result do not give us much understanding of the structure of the cryptogenography problem. We therefore imagine that a better understanding of this smallest-possible problem of leaking one bit from two players, ideally by determining an optimal protocol (that is, matching a hardness result), could greatly improve the situation.

Our Results. We shall be partially successful in achieving these goals. On the positive side, we find protocols with strictly larger success probability than $\frac{1}{3}$ (namely 0.3384) and we prove a stricter hardness result of 0.3672. Our new protocols look very different from the 2-round protocol given by Brody et al., in particular, they use infinite protocol trees (but have an expected finite number of communication rounds). These findings motivate and give new starting points for further research on the cryptogenography problem.

On the not so positive side, our work on better protocols indicates that good cryptogeno-graphic protocols can be very complicated. The simplest protocol we found that beats the $\frac{1}{3}$

barrier already has a protocol tree of depth 16, that is, the two players need to communicate for 16 rounds in the worst case. While we still manage to give a human-readable description and performance proof for this protocol, it is not surprising that previous works not incorporating a computer-assisted search did not find such a protocol. Our best protocol, giving a success probability of 0.3384, already uses 18248 non-equivalent states.

Technical contributions. To find the improved protocols, we use a number of theoretical and experimental tools. We first reformulate the cryptogenography problem as a solitaire vector splitting game over vectors in $\mathbb{R}_{\geq 0}^{2\times k}$. Both for human researchers and for automated protocol searches, this reformulation seems to be easier to work with than the previous reformulation via convex combinations of distributions lying in a common allowed plane [2]. It also proved to be beneficial for improving upon the hardness result.

Restrictions of the vector splitting game to a finite subset of $\mathbb{R}_{\geq 0}^{2\times k}$, e.g., $\{0, \dots, T\}^{2\times k}$, can easily be solved via dynamic programming, giving (due to the restriction possibly suboptimal) cryptogenographic protocols. Unfortunately, for $k = 2$ even discretizations as fine as $T = 40$ are not sufficient to find protocols beating the 1/3 barrier and memory usage quickly becomes a bottleneck issue. However, exploiting the simple fact that the game values are homogeneous (that is, multiplying a game position by a non-negative scalar changes the game value by this factor), we can (partially) simulate a much finer discretization in a coarse one. This *extended dynamic programming approach* easily gives cryptogenographic success probabilities larger than 1/3. Reading off the corresponding protocols, due to the reuse of the same position in different contexts, needs more care, but in the end gives without greater difficulties also the improved protocols.

When a cryptogenographic protocol reuses a state a second time (with a non-trivial split in between), then there is no reason to re-iterate this part of the protocol whenever this position occurs. Such a protocol allows infinite paths, while still needing only an expected finite number of rounds. Since the extended dynamic programming approach in finite time cannot find such protocols, we use a linear programming based post-processing stage. We translate each splitting operation used in the extended dynamic programming search into an inequality relating game values. By exporting these into an LP-solver, we do not only obtain better game values (possibly corresponding to cryptogenographic protocols with infinite paths, for which we would get a compact representation by making the cycles explicit), but also a way to easily certify these values using an optimality check for a linear program instead of having to trust the ad-hoc dynamic programming implementation.

Related work. Despite a visible interest of the research community in the cryptogenography problem, the only relevant follow-up work is Jakobsen's paper [5], which analyses the cryptogenography problem for the case that several of the players know the secret. This allows to leak a much larger amount of information as made precise in [5]. Due to the asymptotic nature of these results, unfortunately, they do not give new insight in the 2-player case. Other work on anonymous broadcasting typically assumes bounded computational power of the adversary (see, e.g., [6]); we refer to [3] for a survey on anonymous communication in communication networks.

In [2], the cryptogenography problem was reformulated to the problem of finding the point-wise minimal function f on the set of secret distributions that is point-wise not smaller than some given function g and that is concave on an infinite set of 2-dimensional planes. Such restricted notions of concavity (or, equivalently, convexity) seem to be less understood. We found work by Matoušek [9] for a similar convexity problem, however, with only a finite number of one-dimensional directions in which convexity is required. We do not see how to extend these results to our needs.

2 Finding Better Cryptogenography Protocols

This section is devoted to the design of stronger cryptogenographic protocols. In particular, we demonstrate that a success probability of more than 1/3 can be achieved. We start by making the cryptogenography problem precise (Section 2.1) and introduce an equivalent formulation as solitaire vector splitting game (Section 2.2). We illustrate both formulations using the best known protocol for the 2-player case (Section 2.3). In Section 2.4, we state basic properties that simplify the analysis of protocols and aid our automated search for better protocols, which is detailed in Section 2.6. In Section 2.5, we give a simple, human-readable proof that 1/3 is not the optimal success probability by analyzing a protocol with success probability $\frac{449}{1334} \approx 0.3341$. We describe how to post-optimize and certify the results obtained by the automated search using linear programming in Section 2.7 and summarize our findings (in particular, the best lower bound we have found) in Section 2.8. Proofs and details that had to be omitted due to space constraints can be found in an extended version of this article [4].

2.1 The Cryptogenography Problem

Let us fix an arbitrary number k of players called $1, \dots, k$ for simplicity. We write $[k] := \{1, \dots, k\}$ for the set of players. We assume that a random one of the them, the "secret owner" $J \in [k]$, initially has a secret, namely a random bit $X \in \{0, 1\}$. The task of the players is, using public communication only, to make this random bit public without revealing the identity of the secret owner. More precisely, we assume that the players, before looking at the secret distribution, (publicly) decide on a communication protocol π. This is again public, that is, all bits sent are broadcast to all players, and they may depend only on previous communication, the private knowledge of the sender (whether he is the secret owner or not, and if so, the secret), and private random numbers of the sender. At the end of the communication phase, the protocol specifies an output bit Y (depending on all communication).

The aspect of not disclosing the identity of the secret owner is modeled by an adversary, who knows the protocol (because it was discussed in public) and who gets to see all communication (and consequently also knows the protocol output Y). The adversary, based on all this data, blames one player K. The players win this game if the protocol outputs the true secret (that is, $Y = X$) and the adversary does not blame the secret owner (that is, $K \neq J$), otherwise the adversary wins. It is easy to see what the best strategy for the adversary is (given the protocol and the communication), so the interesting part of the cryptogenography problem is finding strategies that maximize the probability that the players win assuming that the adversary plays optimally. We call this the (players') success probability of the protocol.

While the game starts with a uniform secret distribution, it will be useful to regard arbitrary secret distributions. In general, a *secret distribution* is a distribution D over $\{0, 1\} \times [k]$, where D_{ij} is the probability that player $j \in [k]$ is the secret owner and the secret is $i \in \{0, 1\}$. Modulo a trivial isomorphism, D is just a vector in $\mathbb{R}_{\geq 0}^{2 \times k}$ with $\|D\|_1 = 1$. We denote by $\Delta = \Delta_k$ the set of all these distributions (this was denoted by $\Delta(\{0, 1\} \times [k])$ in [2]).

Brody et al. [2] observe that any cryptogenographic protocol can be viewed as successive rounds of one-bit communication, where in each step some (a priori) secret distribution probabilistically leads to one of two follow-up (a posteriori) distributions (depending on the bit transmitted) such that the a priori distribution is a convex combination of these and a certain proportionality condition is fulfilled (all three distributions lie in the same "allowed plane"). Conversely, whenever the initial distribution can be written as such a convex combination of certain distributions, then there is a round of a cryptogenographic protocol leading to these two distributions (with certain probabilities). Consequently, the problem of finding a good cryptogenographic protocol is equivalent to iteratively rewriting the initial equidistribution as certain convex combinations of other secret distributions in such a way that the success probability, which can be expressed in terms of this rewriting tree, is large.

2.2 The Solitaire Vector Splitting Game

Instead of directly using the "convex combination" formulation of Brody et al., we propose a slightly different reformulation as *solitaire vector splitting game.* This formulation seems to ease finding good cryptogenographic protocols (lower bounds for the success probability), both for human researchers and via automated search (Section 2.5). The main advantage of our formulation is that it takes as positions all $2k$-dimensional vectors with non-negative entries, whereas the cryptogenographic protocols are only defined on distributions over $\{0,1\} \times [k]$. In this way, we avoid arguing about probabilities and convex combinations and instead simply split a vector (resembling a secret distribution) into a sum of two other vectors. Furthermore, a simple monotonicity property (Proposition 2.5) eases the analyses. Still, there is an easy translation between the two formulations, so that we can re-use whatever results were found in [2].

For reasons of space, we do not repeat in detail the "convex combination" formulation and its equivalence to cryptogenographic protocols (the latter alone takes around two pages in [2]). We focus instead on the formal introduction of the vector splitting game (and argue that its equivalent to the "convex combination" formulation in Lemma 2.6) and refer to [2, 4] for a more detailed exposition.

▶ **Definition 2.1.** Let $D \in \mathbb{R}_{\geq 0}^{2\times k}$. We say that (D_0, D_1) is a *j-allowed split of D* if $D = D_0 + D_1$ and D_0 (and thus also D_1) is proportional to D on $\{0,1\} \times ([k] \setminus \{j\})$, that is, there is a $\lambda \in [0,1]$ such that $(D_0)_{|\{0,1\}\times([k]\setminus\{j\})} = \lambda D_{|\{0,1\}\times([k]\setminus\{j\})}$. We call (D_0, D_1) an *allowed split* of D if it is a j-allowed split of D for some $j \in [k]$.

The objective of the vector splitting game is to recursively apply allowed splits to a given vector $D \in \mathbb{R}_{\geq 0}^{2\times k}$ with the target of maximizing the sum of the

$$p(D') := \max_{x\in\{0,1\}} \left(\sum_{j\in[k]} D'_{x,j} - \max_{j\in[k]} D'_{x,j} \right)$$

values of the resulting vectors (note that when D' is a distribution, then $p(D')$ is the 0-bit success probability of D'; for more details, the reader is referred to [2, 4]). More precisely, an n-round play of the vector splitting game is described by a binary tree of height at most n, where the nodes are labeled with *game positions* in $\mathbb{R}_{\geq 0}^{2\times k}$. The root is labeled with the initial position D. For each non-leaf v, the labels of the two children form an allowed split of the label of v. The payoff of such a play is $\sum_{D'} p(D')$, where D' runs over all leaves of the game tree. The aim is to maximize the payoff. Right from this definition, it is clear that the maximum payoff achievable in an n-round game started in position D, the *value* of this game, is $\textsc{succ}_n(D)$ as defined below.

▶ **Definition 2.2.** For all $n \in \mathbb{N}$ and for all $D \in \mathbb{R}^{2\times k}_{\geq 0}$, we recursively define

(i) $\textsc{succ}_0(D) := \max\limits_{x\in\{0,1\}} \left(\sum\limits_{j\in[k]} D_{x,j} - \max\limits_{j\in[k]} D_{x,j} \right)$;

(ii) $\textsc{succ}_n(D) := \max\limits_{(D_0,D_1)} \Big(\textsc{succ}_{n-1}(D_0) + \textsc{succ}_{n-1}(D_1) \Big)$, if $n \geq 1$. Here the maximum is taken over all allowed splits (D_0, D_1) of D.

For an example of an admissible game, we refer to Figure 1 in Section 2.3.

It is easy to see that the game values are non-decreasing in the number of rounds, but bounded. The limiting value is thus well-defined.

▶ **Lemma 2.3.** *Let $D \in \mathbb{R}^{2\times k}_{\geq 0}$ and $n \in \mathbb{N}$. Then $\textsc{succ}_n(D) \leq \|D\|_1$ and $\textsc{succ}_{n+1}(D) \geq \textsc{succ}_n(D)$. Consequently, $\textsc{succ}(D) := \lim_{n\to\infty} \textsc{succ}_n(D)$ is well-defined and is equal to $\sup_{n\in\mathbb{N}} \textsc{succ}_n(D)$.*

▶ **Proposition 2.4** (scalability)**.** *Let $D \in \mathbb{R}^{2\times k}_{\geq 0}$ and $\lambda \geq 0$. Then $\textsc{succ}_n(\lambda D) = \lambda\, \textsc{succ}_n(D)$ for all $n \in \mathbb{N}$. Consequently, $\textsc{succ}(\lambda D) = \lambda\, \textsc{succ}(D)$.*

▶ **Proposition 2.5** (monotonicity)**.** *Let $D, E \in \mathbb{R}^{2\times k}_{\geq 0}$ with $E \geq D$ (component-wise). Then $\textsc{succ}_n(E) \geq \textsc{succ}_n(D)$ for all $n \in \mathbb{N}$. Consequently, $\textsc{succ}(E) \geq \textsc{succ}(D)$.*

From the previous definitions and observations, we derive that the game values for games starting with a distribution D, that is, $\|D\|_1 = 1$, and the success probabilities of the optimal cryptogenographic protocols for D, are equal.

▶ **Lemma 2.6.** *Let $D \in \mathbb{R}^{2\times k}_{\geq 0}$ with $\|D\|_1 = 1$. Then for all $n \in \mathbb{N}$, our definitions of $\textsc{succ}_n$ coincide with the ones of Brody et al., which are the success probabilities of the best n-round cryptogenographic protocols for the distribution D. Consequently, also the definition of $\textsc{succ}(D)$ coincides.*

2.3 Example: The Best-so-far 2-Player Protocol

We now turn to the case of two players. We use this subsection to describe the best known protocol for two players in the different languages. We also use this mini-example to sketch the approaches used in the following subsections to design superior protocols.

For two players, we usually write a game position $D = (D_{01}, D_{11}, D_{02}, D_{12}) \in \mathbb{R}^{2\times 2}_{\geq 0}$ as $D = (a, b, c, d)$. The 0-round game value (equaling the success probability of the 0-bit protocol) then is

$$\textsc{succ}_0(D) = \max\{\min\{a, c\}, \min\{b, d\}\}.$$

As a warmup, let us describe the best known 2-player protocol TwoBit in the two languages. In the language of Brody et al., the first player can send a (randomized) bit that transforms the initial distribution $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ with probability $\frac{1}{2}$ each into the distributions $(\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6})$ and $(\frac{1}{6}, \frac{1}{6}, \frac{1}{3}, \frac{1}{3})$. In the first case, the second player can send a bit leading to each of the distributions $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0)$ and $(\frac{1}{3}, \frac{1}{3}, 0, \frac{1}{3})$ with probability $\frac{1}{2}$, both having a 0-bit success probability of $\frac{1}{3}$. In the second possible result of the first move, the first player can lead to an analogous situation. Consequently, after two rounds of the protocol we end up with four equally likely distributions all having a 0-bit success probability of $\frac{1}{3}$. Hence the protocol TwoBit has a success probability of $\frac{1}{3}$.

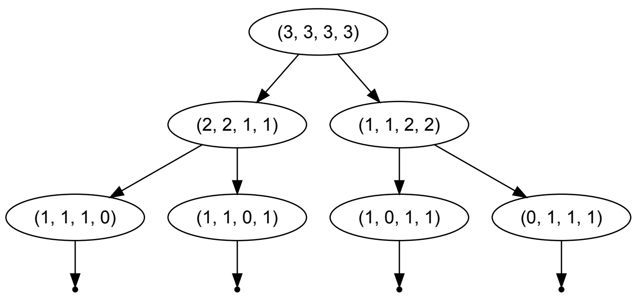

Figure 1 Game tree corresponding to TwoBit.

In the language of the splitting games, we can forget about the probabilities and simply split up the initial distribution. Using the scaling invariance, to ease reading we scaled up all numbers by a factor of 12. Figure 1 shows the game tree corresponding to the TwoBit protocol. It shows that $\textsc{succ}_2(3,3,3,3) \geq 4$, proving again the existence of a cryptogenographic protocol for the distribution $(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) = \frac{1}{12}(3,3,3,3)$ with success probability $\frac{4}{12} = \frac{1}{3}$.

Note that each allowed split (D^0, D^1) of D implies the inequality $\textsc{succ}(D) \geq \textsc{succ}(D^0) + \textsc{succ}(D^1)$, which follows from clause (ii) of Definition 2.2 and taking the limit $n \to \infty$. Hence the game tree giving the $\frac{1}{3}$ lower bound for the success probability equivalently gives the following proof via inequalities.

$$\begin{aligned} \textsc{succ}(3,3,3,3) &\geq \textsc{succ}(2,2,1,1) + \textsc{succ}(1,1,2,2), \\ \textsc{succ}(2,2,1,1) = \textsc{succ}(1,1,2,2) &\geq \textsc{succ}(1,0,1,1) + \textsc{succ}(0,1,1,1), \\ \textsc{succ}(1,0,1,1) = \textsc{succ}(0,1,1,1) &\geq \textsc{succ}_0(0,1,1,1) = 1. \end{aligned}$$

The splitting game and the inequality view will in the following be used to design stronger protocols (better lower bounds for the optimal success probability). We shall compute good game trees by computing lower bounds for the game values of a discrete set of positions via repeatedly trying allowed splits. For example, the above game tree for the starting position $(3,3,3,3)$ could have easily be found by recursively computing the game values for all positions in $\{0,1,2,3\}^4$.

It turns out that such an automated search leads to better results when we also allow scaling moves (referring to Proposition 2.4). For example, in the above mini-example of computing optimal game values for all positions $\{0,1,2,3\}^4$, we could try to exploit the fact that $\textsc{succ}(1,1,1,1) = \frac{1}{3}\textsc{succ}(3,3,3,3)$. Such scaling moves are a cheap way of working in $\{0,1,2,3\}^4$ while trying to gain the power of working in $\{0,1,\ldots,9\}^4$, which would be computationally more costly, especially with regard to memory usage. Scaling moves may lead to repeated visits of the same position, resulting in cyclic structures. Here translating the allowed splits used in the tree into the inequality formulation and then using an LP-solver is an interesting approach (detailed in Section 2.7). It allows to post-optimize the game trees found, in particular, by solving cyclic dependencies. This leads to slightly better game values and compacter representations of game trees.

2.4 Useful Facts

For some positions of the vector splitting game, the true value is easy to determine. We do this here to later ease the presentation of the protocols.

▶ **Proposition 2.7.** *We have* $\textsc{succ}(a,b,c,d) \le \min\{a,c\} + \min\{b,d\}$.

▶ **Proposition 2.8.** *Let* $D = (a,b,c,0)$. *Then* $\textsc{succ}(D) = \textsc{succ}_0(D) = \min\{a,c\}$.

▶ **Proposition 2.9.** *If* $D = (a,b,c,d)$ *is such that* $a + b \le \min\{c,d\}$, *then* $\textsc{succ}(D) = a + b$.

2.5 Small Protocols Beating the $1/3$ Barrier

We now present a sequence of protocols showing that there are cryptogenographic protocols having a success probability strictly larger than $\frac{1}{3}$. These protocols are still relatively simple, so we also obtain a human-readable proof of the following result.

▶ **Theorem 2.10.** $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) \ge \frac{449}{1334} \approx 0.3341$.

Proof. To be able to give a readable mathematical proof, we argue via inequalities for game values $\textsc{succ}(\cdot)$. We later discuss how the corresponding protocols (game trees) look like.

We first observe the following inequalities, always stemming from allowed splits (the underlined entries are proportional). Whenever Proposition 2.8 or 2.9 determine a value, we exploit this without further notice.

$$\begin{aligned}\textsc{succ}(\underline{12,12},12,12) &\ge \textsc{succ}(\underline{7,7},6,4) + \textsc{succ}(\underline{5,5},6,8),\\ \textsc{succ}(\underline{5,5},6,8) &\ge \textsc{succ}(\underline{2,2},0,2) + \textsc{succ}(\underline{3,3},6,6) = 2 + 6 = 8.\end{aligned}$$

This proves $\textsc{succ}(12,12,12,12) \ge 8 + \textsc{succ}(7,7,6,4)$. To analyze $\textsc{succ}(7,7,6,4)$, we use the allowed split

$$\textsc{succ}(7,7,\underline{6,4}) \ge \textsc{succ}(4,5,\underline{3,2}) + \textsc{succ}(3,2,\underline{3,2}) \tag{1}$$

and regard the two positions $(4,5,3,2)$ and $(3,2,3,2)$ separately in some detail.

Claim 1: The value of $(4,5,3,2)$ satisfies succ$(4,5,3,2) \ge \frac{55}{12}$. By scaling, we have $\textsc{succ}(4,5,3,2) = \frac{1}{2}\textsc{succ}(8,10,6,4)$. We present the allowed splits

$$\begin{aligned}\textsc{succ}(\underline{8,10},6,4) &\ge \textsc{succ}(\underline{4,5},2,4) + \textsc{succ}(\underline{4,5},4,0) = \textsc{succ}(4,5,2,4) + 4,\\ \textsc{succ}(4,5,\underline{2,4}) &\ge \textsc{succ}(1,2,\underline{1,2}) + \textsc{succ}(3,3,\underline{1,2}) = \textsc{succ}(1,2,1,2) + 3,\end{aligned}$$

hence $\textsc{succ}(8,10,6,4) \ge \textsc{succ}(1,2,1,2) + 7$. To bound the latter term, we use the scaling $\textsc{succ}(1,2,1,2) = \frac{1}{6}\textsc{succ}(6,12,6,12)$ and consider the allowed splits

$$\begin{aligned}\textsc{succ}(\underline{6,12},6,12) &\ge \textsc{succ}(\underline{5,10},3,9) + \textsc{succ}(\underline{1,2},3,3) = \textsc{succ}(5,10,3,9) + 3,\\ \textsc{succ}(5,10,\underline{3,9}) &\ge \textsc{succ}(0,6,\underline{2,6}) + \textsc{succ}(5,4,\underline{1,3}) = 6 + 4 = 10.\end{aligned}$$

Thus $\textsc{succ}(6,12,6,12) \ge 13$ and $\textsc{succ}(1,2,1,2) \ge \frac{13}{6}$. This shows $\textsc{succ}(4,5,3,2) \ge \frac{1}{2}(\textsc{succ}(1,2,1,2) + 7) \ge \frac{55}{12}$.

Claim 2: We have succ$(3,2,3,2) \ge \frac{5}{3} + \frac{2}{9}$succ$(7,7,6,4)$. By scaling, we obtain $\textsc{succ}(3,2,3,2) = \frac{1}{3}\textsc{succ}(9,6,9,6)$ and compute

$$\begin{aligned}\textsc{succ}(9,6,\underline{9,6}) &\ge \textsc{succ}(6,3,\underline{6,4}) + \textsc{succ}(3,3,\underline{3,2}),\\ \textsc{succ}(6,3,\underline{6,4}) &\ge \textsc{succ}(3,0,\underline{3,2}) + \textsc{succ}(3,3,\underline{3,2}) = 3 + \textsc{succ}(3,3,3,2),\end{aligned}$$

and hence $\textsc{succ}(9,6,9,6) \ge 3 + 2\,\textsc{succ}(3,3,3,2)$. To bound the latter term, we scale $\textsc{succ}(3,3,3,2) = \frac{1}{3}\textsc{succ}(9,9,9,6)$ and present the allowed splits

$$\begin{aligned}\textsc{succ}(\underline{9,9},9,6) &\ge \textsc{succ}(\underline{7,7},6,4) + \textsc{succ}(\underline{2,2},3,2),\\ \textsc{succ}(\underline{2,2},3,2) &\ge \textsc{succ}(\underline{1,1},1,0) + \textsc{succ}(\underline{1,1},2,2) = 1 + 2 = 3.\end{aligned}$$

Thus $\textsc{succ}(3,3,3,2) \ge 1 + \frac{1}{3}\textsc{succ}(7,7,6,4)$, implying $\textsc{succ}(3,2,3,2) \ge \frac{5}{3} + \frac{2}{9}\textsc{succ}(7,7,6,4)$.

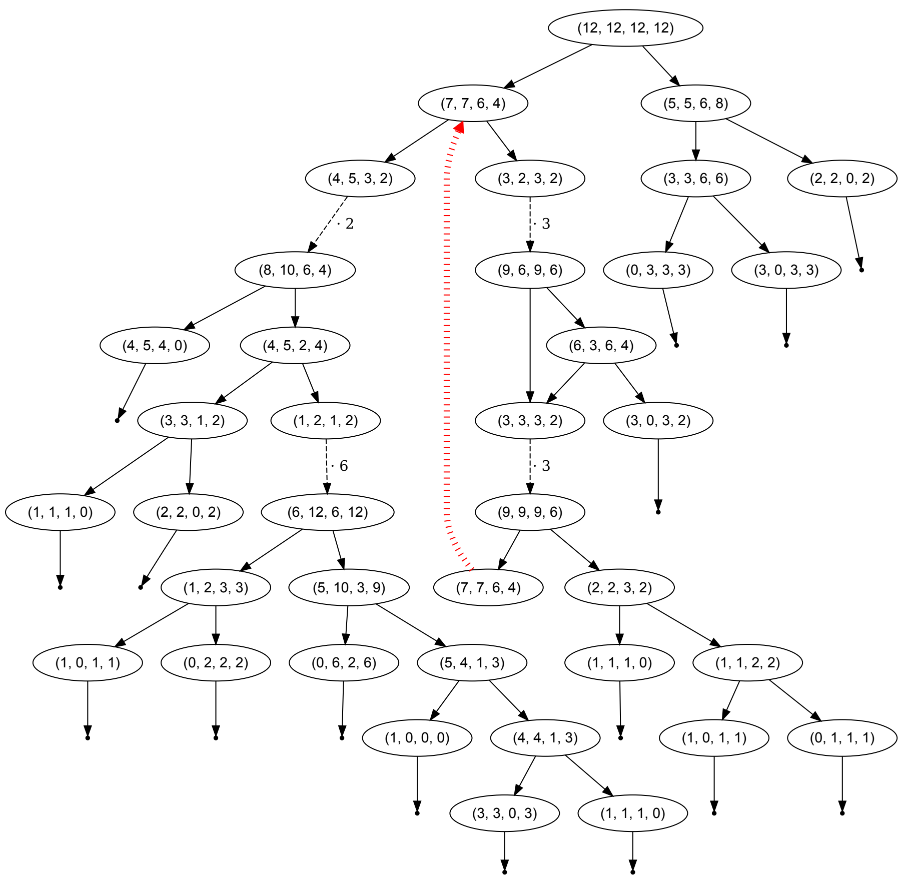

Figure 2 Game tree representation of the protocols of Theorem 2.10.

Putting things together. Claims 1 and 2 together with (1) give

$$\textsc{succ}(7,7,6,4) \geq \tfrac{75}{12} + \tfrac{2}{9}\,\textsc{succ}(7,7,6,4).$$

By solving this elementary equation, we obtain $\textsc{succ}(7,7,6,4) \geq \frac{225}{28}$, and it follows that $\textsc{succ}(12,12,12,12) \geq \frac{225}{28} + 8 = \frac{449}{28} = 16 + \frac{1}{28}$. Scaling leads to the claim of the theorem $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) \geq \frac{1}{3} + \frac{1}{1344} = \frac{449}{1344} = 0.33407738\ldots$. ◀

When translating the inequalities into a game tree (see Figure 2 for the result), we first observe that in Claim 2 we obtained two different nodes labeled with the position $(3,3,3,2)$. Since there is no reason to treat them differently, we can identify these two nodes and thus obtain a more compact representation of the game tree. This is the reason why the node labeled $(3,3,3,2)$ in Figure 2 has two incoming edges.

Interestingly, such identifications can lead to cycles. If we translate the equations for position $(7,7,6,4)$ and its children into a graph, then we observe that the node for $(7,7,6,4)$ has a descendant also labeled $(7,7,6,4)$ (this is what led to the inequality $\textsc{succ}(7,7,6,4) \geq \frac{75}{12} + \frac{2}{9}\,\textsc{succ}(7,7,6,4)$). By transforming this inequality to $\textsc{succ}(7,7,6,4) \geq \frac{225}{28}$, we obtain a statement that is true, but that does not anymore refer to an actual (finite) game tree. However, there is a sequence of game trees with values converging to the value we determined. These trees are obtained from recursively applying the above splitting procedure for $(7,7,6,4)$ a certain number ℓ of times and then using the 0-round tree for the lowest node

Table 1 Lower bounds $s(T,\dots,T)/(4T)$ on $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})$ stemming from the automated search (line 1). Given are also the number of iterations until the automated search procedure converged, i.e., stopped finding improvements using allowed splits or scalings, and the number of game positions and constraints that had an influence on the value of $s(T,\dots,T)$.

T	15	20	25	30
Automated search	0.3369432925	0.3376146092	0.3379027186	0.3381689066
Iterations	119	129	141	146
Constraints	535	1756	4217	13958
Game Positions	394	1326	2956	9646

labeled $(7,7,6,4)$. The value of this game tree is $8+\sum_{i=0}^{\ell-1}(\frac{2}{9})^i\frac{75}{12}+(\frac{2}{9})^\ell\,\textsc{succ}_0(7,7,6,4) = 8+\frac{75}{12}\frac{1-(\frac{2}{9})^\ell}{1-\frac{2}{9}}+6\cdot(\frac{2}{9})^\ell = \frac{449}{28}-\frac{57}{28}\,(\frac{2}{9})^\ell$. Hence for $\ell \geq 3$, this is more than 16 (which represents a success probability of $\frac{1}{3}$), corresponding to a game tree of height[1] $4+4\ell \geq 16$.

2.6 Automated Search

The vector splitting game formulation allows to search for good cryptogenographic protocols as follows. We try to determine the game values of all positions from a discrete set $\mathcal{D} := \{0,\dots,T\}^{2\times k}$ by repeatedly applying allowed splits. More precisely, we store a function $s:\mathcal{D}\to\mathbb{R}$ that gives a lower bound on the game value $\textsc{succ}(D)$ of each position $D\in\mathcal{D}$. We initialize this function with $s\equiv\textsc{succ}_0$ and then in order of ascending $\|D\|_1$ try all allowed splits $D=D_0+D_1$ and update $s(D)\leftarrow s(D_0)+s(D_1)$ in case we find that $s(D)$ was smaller.

Recall that for any secret distribution D, the game value $\textsc{succ}(D)$ is the supremum success probability of cryptogenographic protocols for D. Hence, e.g., the value $s(T,\dots,T)/(2Tk) \leq \textsc{succ}(1/(2k),\dots,1/(2k))$ is a lower bound for this success probability. As we will discuss later, by keeping track of the update operations performed, we can not only compute such a lower bound, but also concrete protocols.

Since even for $k=2$, the size of the position space $\mathcal{D}$ and the number of allowed splits increase quickly with T, only moderate choices of T are computationally feasible, limiting the power of this approach drastically. However, using the scaling invariance $\lambda\textsc{succ}(D) = \textsc{succ}(\lambda D)$, we can introduce a scaling step: we iteratively optimize using allowed splits[2], then backpropagate the computed values (updating, e.g., $s(1,1,1,1)\leftarrow(1/T)\cdot s(T,T,T,T)$) to repeat the process. Surprisingly, this simple modification is sufficient to find protocols that are better than the previous best protocol TwoBit. For a more precise description of the algorithm, see the extended version of this article [4].

The success probabilities of the protocols computed following the above approach, using different values for T, are given in the first line of Table 1. Further results exploiting the post-optimization are given in Table 2 in Section 2.8.

2.7 Post-Optimization via Linear Programming

When letting the automated search also keep track of at what time which update operation was performed, this data can be used to extract strategies for the splitting game (and

[1] Note that the height refers to the number of transmitted bits and thus does not include the number of (virtual) scaling moves.

[2] In fact, we use *relaxed* splits, in which some coordinate in the resulting distributions may additionally be rounded down – this slightly increases the set of admissible splits. For details, see [4].

Table 2 Lower bounds $s(T,\dots,T)/(4T)$ on $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})$ stemming from the automated search only (line 1) and from the LP solution of the linear system extracted from the automated search data (line 2), when the number of iterations is restricted to 20.

T	30	35	40	45	50
Autom. search	0.3381086510	0.3381937725	0.3383218072	0.3383946540	0.3384414508
LP solution	0.3381527322	0.3382301900	0.3383547901	0.3384303130	0.3384736461
Iterations	20	20	20	20	20
Constraints	5373	8882	12410	18659	24483
Game states	4126	6789	9396	13992	18248

cryptogenographic protocols). Some care has to be taken to only extract those intermediate positions that had an influence on the final game value for the position we are interested in.

While this approach does deliver good cryptogenographic protocols, manually verifying the correctness of the updates or analyzing the structure of the underlying protocol quickly becomes a difficult task, as the size of the protocol grows rapidly. Fortunately, it is possible to output a compact, machine-verifiable certificate for the lower bound obtained by the automated search that might even prove a better lower bound than computed: Each update step in the automated search corresponds to a valid inequality of the form $\textsc{succ}(D) \geq \textsc{succ}(D_0)+\textsc{succ}(D_1)$, $\textsc{succ}(D) \geq \textsc{succ}_0(D)$ or $\lambda\cdot\textsc{succ}(D) = \textsc{succ}(\lambda\cdot D)$. We can extract the (sparse) set $\mathrm{ineq}(T,T,T,T)$ of those inequalities that lead to the computed lower bound on $\textsc{succ}(T,T,T,T)$.

Consider replacing each occurrence of $\textsc{succ}(D')$ in the set of inequalities $\mathrm{ineq}(T,T,T,T)$ found by the automated search by a variable $s_{D'}$. We obtain a system of linear inequalities S that has the feasible solution $s_{D'} = \textsc{succ}(D')$ (for every occurring vector D'). Hence in particular, the optimal solution of the linear program of *minimizing* s_D *subject to* S is a lower bound on $\textsc{succ}(D)$. It is easy to see that this solution is at least as good as the solution stemming from the automated search alone. It can, however, even be better, in particular when a game strategy yields cyclic visits to certain positions. Table 2 contains, for different values of T, the success probabilities found by automated search (run with a bounded iteration number of 20) and by this above linear programming approach. The table also contains the number of linear inequalities (and game positions) that were extracted from the automated search run. We observe that consistently the LP-based solution is minimally better. We also observe that the number of constraints is still moderate, posing no difficulties for ordinary LP solvers (which stands in stark contrast to feeding all allowed splits and scalings over the complete discretization to the LP solver, which quickly becomes infeasible).

Hence the advantage of our approach of extracting the constraints from the automated search stage is that it generates a much sparser sets of constraints that still are sufficiently meaningful. After solving the LP, we can further sparsify this set of inequalities by deleting all inequalities that are not tight in the optimal solution of the LP, since these cannot correspond to the best splits found for the corresponding vector D, yielding a smaller set of relevant inequalities, which might help to analyze the structure of strong protocols.

2.8 Our Best Protocol

We report the best protocol we found using the approach outlined in the previous sections.

▶ **Theorem 2.11.** *In the 2-player cryptogenography problem,* $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) \geq 0.3384736$.

Proof. On `http://people.mpi-inf.mpg.de/~marvin/verify.html`, we provide a linear program based on feasible inequalities on the discretization $\mathcal{D}$ with $T = 50$. To verify the result, one only has to (1) check validity of each inequality, i.e., checking whether each constraints encodes a feasible scaling, allowed split or zero-bit success probability and (2) solve the linear program. Since we represent the distributions $D = (a, b, c, d)$ using a normal form $a \geq b, c, d$ (to break symmetries), checking validity of each splitting constraint is not completely trivial, but easy. We provide a simple checker program to verify validity of the constraints. The LP is output in a format compatible with the LP solver `lp_solve`[3]. ◀

3 A Stronger Hardness Result

In this section, we prove that any 2-player cryptogenographic protocol has a success probability of at most 0.3672. This improves over the previous 0.375 bound of [2].

▶ **Theorem 3.1.** *We have* $\textsc{succ}(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}) \leq \frac{47}{128} = 0.367188$.

To prove the result, we apply the concavity method used by Brody et al. [2] which consists in finding a function s that (i) is lower bounded by $\textsc{succ}_0$ for all distributions and (ii) satisfies a certain concavity condition. Similar concavity arguments have been applied before in information complexity and information theory (see, e.g., [1, 7, 8]). We first relax the lower bound requirement to hold only for six particular simple distributions (namely $(1, 0, 0, 0), \dots, (0, 0, 0, 1), (\frac{1}{2}, 0, \frac{1}{2}, 0)$ and $(0, \frac{1}{2}, 0, \frac{1}{2})$) instead of all distributions. This simplifies the search for a suitable stronger candidate function satisfying (i) - it remains to verify condition (ii) for the thus found candidate function.

More specifically, we adapt the upper bound function of Brody et al. [2] to

$$\begin{aligned} s(a, b, c, d) &:= \frac{1 - f(a, b, c, d)}{4}, \\ f(a, b, c, d) &:= a^2 + b^2 + c^2 + d^2 - 6(ac + bd) + 8abcd. \end{aligned}$$

In fact, we have changed their upper bound function by introducing an additional term of $8abcd$, which attains a value of zero on the distributions $(\frac{1}{2}, 0, \frac{1}{2}, 0)$, $(1, 0, 0, 0)$, etc., thus not affecting the zero-bit success probability condition of the concavity method. Due to space constraints, we defer the quite technical verification of the concavity condition to [4].

Since this function attains the value of $s(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}) = \frac{47}{128}$ at the uniform distribution, we obtain the stronger upper bound of $\textsc{succ}(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}) \leq \frac{47}{128}$ of Theorem 3.1.

4 Conclusion

Despite the fundamental understanding of the cryptogenography problem obtained by Brody et al. [2], determining the success probability even of the 2-player case remains an intriguing open problem. The previous best protocol with success probability 1/3, while surprising and unexpected at first, is natural and very symmetric (in particular when viewed in the vector splitting game formulation). We disprove the hope that it is an optimal protocol by exhibiting less intuitive and less symmetric protocols having success probabilities up to 0.3384. Concerning hardness results, our upper bound of 0.3671875 shows that also the previous upper bound of 3/8 was not the final answer. These findings add to the impression that the cryptography problem offers a more complex nature than its simple description might suggest and that understanding the structure of good protocols is highly non-trivial.

[3] `http://lpsolve.sourceforge.net`

We are optimistic that our methods support a further development of improved protocols and bounds. (1) Trivially, investing more computational power or optimizing the automated search might lead to finding better protocols. (2) Our improved protocols might motivate to (manually) find infinite protocol families exploiting implicit properties and structure of these protocols. (3) Our reformulations, e.g., as vector splitting game, might ease further searches for better protocols and for better candidate functions for a hardness proof.

References

1 Mark Braverman, Ankit Garg, Denis Pankratov, and Omri Weinstein. From information to exact communication. In *Proc. 45th Annual ACM Symposium on Theory of Computing (STOC'13)*, pages 151–160. ACM, 2013.

2 Joshua Brody, Sune K. Jakobsen, Dominik Scheder, and Peter Winkler. Cryptogenography. In *Proc. 5th Conference on Innovations in Theoretical Computer Science (ITCS'14)*, pages 13–22, 2014.

3 George Danezis and Claudia Diaz. A survey of anonymous communication channels. Technical Report MSR-TR-2008-35, Microsoft Research, January 2008.

4 Benjamin Doerr and Marvin Künnemann. Improved Protocols and Hardness Results for the Two-Player Cryptogenography Problem. *ArXiv e-prints*, 2016. `arXiv:1603.06113`.

5 Sune K. Jakobsen. Information theoretical cryptogenography. In *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP'14)*, volume 8572 of *Lecture Notes in Computer Science*, pages 676–688. Springer, 2014.

6 Sune K. Jakobsen and Claudio Orlandi. How to bootstrap anonymous communication. In *Proc. 7th Conference on Innovations in Theoretical Computer Science (ITCS'16)*, pages 333–344, 2016.

7 Nan Ma and Prakash Ishwar. The infinite-message limit of two-terminal interactive source coding. *IEEE Trans. Information Theory*, 59(7):4071–4094, 2013.

8 Nan Ma, Prakash Ishwar, and Piyush Gupta. Interactive source coding for function computation in collocated networks. *IEEE Trans. Information Theory*, 58(7):4289–4305, 2012.

9 Jiří Matoušek. On directional convexity. *Discrete & Computational Geometry*, 25(3):389–403, 2001.