Improving the security of device-independent weak coin flipping

Atul[*] Jamie Sikora[†] Tom[‡]

May 11, 2021

1 Introduction

Coin-flipping is the two-party cryptographic primitive where two parties, henceforth called Alice and Bob, wish to generate a random coin-flip and, to make things interesting, they do not trust each other. This primitive was introduced by Blum [**?**] who also introduced the first (classical) protocol. Since then, a series of quantum protocols were introduced which kept improving the security. Mochon finally settled the question about the limits of the security in the quantum regime by proving the existence protocols with security approaching the ideal limit [**?**]. Mochon's work was based on the notion of point games, a concept introduced by Kitaev. Since then, a sequence of works have studied point games and Reference [**?**]. In particular, the proof has been simplified [**?**] and made explicit [**?**]. Interestingly, Miller [**?**] used Mochon's proof to show that protocols approaching the ideal limit must have an exponentially increasing number of messages. We note that all of this work is in the *device-dependent* setting where *Alice and Bob trust their quantum devices*. In this work, we *revise* the security definitions such that when Alice or Bob cheat, they have control of each other's quantum devices, opening up a plethora of new cheating strategies that were not considered in the previously mentioned references.

In this paper, we mostly consider *weak* coin flipping (WCF) protocols. The prefix *weak* refers to the situation where Alice and Bob desire opposite outcomes of the coin. When designing weak coin flipping protocols, the security goals are:

Completeness for honest parties:	If Alice and Bob are honest, then they share the same outcome of a protocol $c \in \{0,1\}$, and c is generated uniformly at random by the protocol.
Soundness against cheating Bob:	If Alice is honest, then a dishonest (i.e., cheating) Bob cannot force the outcome $c = 1$.
Soundness against cheating Alice:	If Bob is honest, then a dishonest (i.e., cheating) Alice cannot force the outcome $c = 0$.

The commonly adopted goal of two-party protocol design is to assume perfect completeness and then minimize the effects of a cheating party, i.e., to make it as sounds as possible. This way, if no parties cheat, then the protocol at least does what it is meant to still. With this in mind, we need a means to measure of the effects of a cheating party. It is often convenient to have a single measure to determine if one protocol is better than another. For these, we use *cheating probabilities* (denoted P_B^* and P_A^*) and *bias* (denoted ε), defined as

[*]California Institute of Technology ????
[†]Virginia Polytechnic Institute and State University. sikora@vt.edu
[‡]Where are you working now??? ?????

P_B^*: The maximum probability with which a dishonest Bob can force an honest Alice to accept the outcome $c = 1$.

P_A^*: The maximum probability with which a dishonest Alice can force an honest Bob to accept the outcome $c = 0$.

ε: The maximum amount with which a dishonest party can bias the probability of the outcome away from uniform. Explicitly, $\varepsilon = \max\{P_B^*, P_A^*\} - 1/2$.

These definitions are not complete in the sense that we have not yet specified how Alice and Bob are capable of. In this work, we study *information theoretic security* meaning that Alice and Bob are only bounded by the laws of quantum mechanics. For example, they are not bounded by polynomial-time quantum computations. In addition to this, we study the security in the *device-independent* regime where we assume Alice and Bob have complete control over the quantum devices when they decide to "cheat".

When studying device-independent (DI) protocols, once should first consider whether or not there are decent classical protocols (since these are not affected by the DI assumption. Indeed, Kitaev [**?**] proved that any classical WCF protocol has bias $\varepsilon = 1/2$, which is the worst possible value. Thus, it makes sense to study quantum WCF protocols in the DI setting, especially if one with bias $\varepsilon < 1/2$ can be found. Indeed, **(Jamie:** author names**)** presented a protocol in [**?**] which has bias $\varepsilon =$????.

In this work, we provide two techniques which can be applied to a wide range of protocols (including [**?**] mentioned above) which can improve the bias. To illustrate our ideas, we now present the protocol in [**?**].

Protocol 1 (DIWCF protocol with $P_A =$??? and $PB =$????? [**?**])**.** *Alice has one box and Bob has two boxes. Each box takes one binary input and gives one binary output and are designed to play the optimal GHZ game strategy. (Who creates and distributes the boxes is not important in the DI setting.)*

1. *Alice chooses a uniformly random input to her box $x \in \{0,1\}$ and obtains the outcome a. She chooses another uniformly random bit $r \in \{0,1\}$ and computes $s = a \oplus (x \cdot r)$. She sends s to Bob.*
2. *Bob chooses a uniformly random bit $g \in \{0,1\}$ and sends it to Alice. (We may think of g as Bob's "guess" for the value of x.)*
3. *Alice sends x and a to Bob. They both compute the output $c = x \oplus g$. This is the outcome of the protocol assuming neither Alice nor Bob abort.*
4. *Bob now tests to see if Alice was honest.*

Test 1*: Bob see if $s = a$ or $s = a \oplus x$. If this is not the case, he knows Alice cheated and aborts.*

Test 2*: Bob chooses a uniformly random bit $y \in \{0,1\}$ and computes $z = x \oplus y \oplus 1$. He inputs y and z into his two boxes and obtains respective outcomes b and c. He aborts if (a, b, c, x, y, z) does not satisfy the winning conditions of the GHZ game.*

To obtain a bias $\varepsilon =$???? protocol from the above, they compose the protocol many times **(Jamie:** Did they discuss how?**)** .

In this work, we build on this protocol using novel pre- and post-processing steps, which we discuss in the next subsection.

1.1 Our main result

We now state the main result of our work.

Theorem 1. *There exists device-independent weak coin flipping protocols with bias approaching* $\varepsilon =$????.

We now discuss how we develop such a protocol. This occurs using two main techniques, *self-testing* and *abort-phobic composition*.

1.1.1 Pre-processing step: Self-testing

In Protocol **??**, a cheating party may control what is in the boxes, both the state and also the mechanics with which the outputs are given. For instance, Bob could **(Jamie:** include details.**)**

We use the concept of self-testing to stop Bob from applying this strategy.

Protocol 2 (Protocol with Alice self-testing)**.** *Alice starts with n boxes, indexed from* 1_1 *to* 1_n*. Bob starts with* $2n$ *boxes, the first half indexed by* 2_1 *to* 2_n *and the last half indexed by* 3_1 *to* 3_n*. The triple of boxes* $(1_i, 2_i, 3_i)$ *is meant to play the optimal GHZ game strategy.*

1. *Alice selects a uniformly random index* $i \in \{1, \ldots, n\}$ *and asks Bob to send her all the boxes* except *those indexed by* 2_i *and* 3_i*.*
2. *Alice plays* $n-1$ *GHZ games using the* $n-1$ *triples of boxes she has, making sure she has a space-like separation between the boxes. (She has long arms.)*
3. *Alice aborts if* any *of the GHZ games fail. Otherwise, she announces to Bob that they can use the remaining boxes for Protocol* **??***.*

The idea is that if n is chosen large enough, then this forces a dishonest Bob to not tamper with the boxes too much. Indeed, this step already allows us to reduce the cheating probabilities.

Lemma 2 (Informal, See Lemma **??** of a formal statement)**.** *When Alice self-tests Bob, the cheating probabilities of Protocol* **??** *in the limit of large n are*

$$\mathrm{P}_A^* =???? \quad \textit{and} \quad \mathrm{P}_B^* =????. \tag{1}$$

To prove this lemma, we have to dive into two technical concepts, which we briefly discuss below.

Rigidity of the GHZ game. We prove that Alice self-tests Bob and passes all $n-1$ plays of the GHZ game, then the remaining triple of boxes has to be approximately performing the optimal GHZ strategy. The differences between this approximation and the optimal strategy disappear in the limit of large n. See Section **??** for details.

Continuity of semidefinite programs. We compute the cheating probabilities using semidefinite programming in the limit of perfect self-testing, as mentioned above. However, we cannot have a protocol with an infinite number of messages. Thus, we study a family of protocols where

the cheating probabilities approach certain thresholds. Thus, we need the semidefinite program values to capture the behaviour of the cheating probabilities as they approach the limit of large n. See Section **??** for details.

Both of these technical steps may find use in independent applications.

1.2 Post-processing step: Abort-phobic composition

Composing WCF protocols is a means to try to balance P_B^* and P_A^* when there is a large difference between them. This effectively reduces the bias ε which may be favourable. To introduce composition, we introduce the notion of *polarity*, which we now define.

Protocol polarity. For a protocol with cheating probabilities satisfying $\mathrm{P}_A^* > \mathrm{P}_B^*$, we say that it has polarity towards Alice. If the cheating probabilities satisfying $\mathrm{P}_B^* > \mathrm{P}_A^*$, we say that it has polarity towards Bob. In either case, we say the protocol is polarized.

Given a polarized protocol P, we may also switch the roles of Alice and Bob since the definition of coin-flipping is symmetric. This switches the polarity of the protocol. It will also be convenient to define P_A to be the version of the protocol with $\mathrm{P}_A^* > \mathrm{P}_B^*$ and P_B to be the version with $\mathrm{P}_B^* > \mathrm{P}_A^*$.

With this in mind, we can now define a simple composition.

Protocol 3 (Winner-gets-polarity composition)**.** *Alice and Bob agree on a protocol P.*

1. *Alice and Bob perform protocol P.*
2. *If Alice won, she polarizes the second protocol towards herself. I.e., they now use the protocol P_A to determine the outcome of the (entire) protocol.*
3. *If Bob won, he polarizes the second protocol towards himself. I.e., they now use the protocol P_B to determine the outcome of the (entire) protocol.*

This is a good way to balance the cheating probabilities of a protocol. For instance, if P has cheating probabilities P_A^* and P_B^* with $\mathrm{P}_A^* > \mathrm{P}_B^*$, then the composition gets to decide "who gets to be Alice" in the second run. We can easily compute Alice's cheating probability in the composition as

$$(\mathrm{P}_A^*)^2 + (1 - \mathrm{P}_A^*)\mathrm{P}_B^* < \mathrm{P}_A^* \tag{2}$$

and Bob's as

$$\mathrm{P}_B^*\mathrm{P}_A^* + (1 - \mathrm{P}_B^*)\mathrm{P}_B^* < \mathrm{P}_A^*. \tag{3}$$

This does indeed reduce the bias.

Abort-phobic composition. In "traditional" way of viewing WCF protocol, there are only two outcomes "Alice wins" (when $c = 0$) or "Bob wins" ($c = 1$). This is because Alice can declare herself the winner if she catches Bob cheating. Similarly, Bob can declare himself the winner if he catches Alice cheating. This is completely fine when we consider "one-shot" versions of these protocols, but we lose something when we compose them. For instance, in the simple composition **??**, Bob should not really accept to continue onto the second protocol if he catches Alice cheating in the first. That is, he knows Alice cheated, so he can declare himself the winner of the entire protocol!

In other word, the equations **??** and **??** may be able to get reduced even further. For purposes of this discussion, suppose Alice adopts a cheating strategy which has a probability α of her winning ($c = 0$), a probability β of her losing ($c = 1$), and a probability γ of Bob detecting Alice cheated. Then her cheating probability in the (abort-phobic) version of the simple composition is now

$$\alpha \cdot \mathrm{P}_A^* + \beta \cdot \mathrm{P}_B^* + \gamma \cdot 0. \tag{4}$$

This quantity may be a strict improvement if $\gamma > 0$ when $\alpha = \mathrm{P}_A^*$.

The concept of abort-phobic composition is simple. Alice and Bob keep using WCF protocols and the winner (at that round) gets to choose the polarity of the subsequent protocol. However, if either party *ever aborts*, then it is game over and the cheating player loses *the entire composition*.

One may think it is tricky to analyze abort-phobic compositions, but we may do this one step at time. To this end, we introduce the concept of *cheat vectors*.

Definition 1 (Alice and Bob's cheat vectors)**.** *Given a protocol, we say that (α, β, γ) is a cheat vector for (dishonest) Alice if there exists a cheating strategy where:*

α is the probability with which Bob accepts the outcome $c = 0$,
β is the probability with which Bob accepts the outcome $c = 1$,
γ is the probability with which Bob aborts.

We can now solve for the optimal bias using *dynamic programming*. Dynamic programming is an optimization tool which provides a mean to solve large optimization problems in smaller steps. For instance, suppose Alice and Bob are now using a protocol midway through the protocol. What Alice must do to optimize her probability of winning *the entire protocol* is to solve the optimization problem

$$\sup_{(\alpha,\beta,\gamma)} \left\{ \begin{array}{rl} & \alpha \cdot \Pr[\text{Alice wins the entire protocol with polarity in the next protocol}] \\ + & \beta \cdot \Pr[\text{Alice wins the entire protocol without polarity in the next protocol}] \\ + & \gamma \cdot 0 \end{array} \right\} \tag{5}$$

where she may choose any strategy with cheat vector (α, β, γ) that she wishes for the current protocol.

Therefore, if we calculate the cheating probabilities from "the bottom up", then we can fix the two probabilities in the expression above and use semidefinite programming to optimize over the cheat vectors.

By using self-testing and abort-phobic compositions, we are able to find protocols which converge onto a bias of $\varepsilon =$???? proving the main result of this work.

1.3 Applications

The concept of polarity extends well beyond finding WCF protocols and, as such, the "winner-gets-polarity" concept allows for WCF to be used in many other compositions. Indeed, we can use it to balance the cheating probabilities in *any* polarized protocol for any symmetric two-party cryptographic task for which such notions can be properly defined.

For instance, many *strong* coin-flipping protocols can be thought of as polarized. For an example, the protocol **??** is indeed a strong coin-flipping protocol. Thus, by balancing the cheating

probabilities of that protocol using our DI WCF protocol and a winner-gets-polarity composition (not even an abort-phobic one!), we get the following theorem.

Theorem 3. *There exists DI strong coin-flipping protocols where no party can cheat with probability greater than* ?????*.*

There are likely more examples of protocols which can be balanced in a DI way using this idea.