Optimally Fair Coin Tossing Protocols

Himanshu Gupta

Abstract—A fair coin flipping protocol attempts to output an unbiased bit in presence of some corrupted party which tries to deviate the expected value of the random bit. This is essentially the basic problem in multiparty computations and has been extensively studied in the last few decades. The best lower bound shown by Cleve [1] has $\Omega(1/r)$ bias in a $r-$round protocol. We follow the approach by Moran [2] and show a variation in the protocol achieves similar bounds on the bias thus proving that Cleve's [1] bound are tight. We further show that such variations in multiparty coin tossing protocols acheive a similar bound in general case.

Index Terms—Randomized Algorithms, Fairness, Coin-flipping protocols, Cryptography

1 Introduction

A fair coin flipping protocol allows the parties to arrive on a common unbiased random bit. The protocol assumes that each of the parties playing the game have not esatblished any trust and rely on the security achieved by the protocol. Such protocols create an important base for many cryptographic applications and have been studied in various different models. However, every model requires formally that the protocol satisfies two important properties:

1) All honest parties output the same bit, chosen uniformly.
2) The presence of corrupted parties should not deviate the output of protocol by significant amount, i.e. the bias should be small.

There are few variants in the protocol based on whether there is an honest majority or not. This gets particulary clear for two-party protocol by Cleve's Impossibility result that there exists an efficient adversary that can bias the output of honest party by $\Omega(1/r)$ in a r-round protocol. Blum's two party protocol [3] relies on the assumption that the corrupt party does not abort for an unbiased output. The case of honest majority forms a special case for multiparty computations [10].
We explore the approach by Moran et al. [2] which constructed a r-round protocol with an optimal bias of $\Theta(1/r)$ and the further work by Beimel [5] which extends this approach to multiparty case.

1.1 Our Contribution

We attempt to show a variation in the approach by Moran [2] , which shows a similar bias without increasing the amount of shares distributed among the parties. Formally, the following theorem is proved.

Theorem 1. *Assuming oblivious transfer exits, there exists an r-round two party coin flipping protocol that is $O(1/4r)$ secure.*

Theorem 2. *For every non-uniform polynomial time adversary A corrupting $t < 2m/3$ parties in a coin tossing protocol described in $CoinToss_3$, the following random variables are $1/p(n) = O(2^3/r)$ close (3 is the number of bits adversary has, in general 2^{k+1}):*

$$SD(IDEAL(1^n), REAL(1^n)) \leq 1/p$$

1.2 Related Work

Coin Flipping. These protocols have been studied in various models with the most notable one being *secure with abort.* In these models the parties do not continue the game once they detect a cheating party or an abort. These are much easier to achieve with optimal security and have been used extensively in other work[[2], [5], [7]]. The approach by Moran [2] has been used as a buidling block in our model also. Initial work by Cleve [1] establishes a lower bound on the security of such protocols and has still not been improved based on the impossibility result.

Further extension by Beimel et al. [5] to a multiparty case presents a general model for an r-round m-party coin flipping protocol based on which we present a 3-party version. These constructions can tolerate up to $2/3$ corrupt parties. The work by Iftach et al. [6] uses a different technique rather than the threshold round which achives an optimal bias of $O(\frac{\log^2 m}{r})$. However this model is restricted to 3 party case and the analysis is much rigorous and complex.

There are other models in the information theoretic setting where the parties have unbounded computational power with public communication channels [9], [10], [11] and in quantum settings [12], [13].

Fairness. A fair secure computation guarantees that if one party recieves the output, then all other parties recieve the same output [14]. Most of the work in this direction uses the technique of a secret round where the output is revealed to the parties and the adversery can only make a guess on when this round is selected. This makes the bias of adversary effective only in case of correct guess and abort in such round. The security of this protocol is formalized by differentiating the execution in an ideal and real model [8].

• *Department of Computer Science, Texas A&M University, College Station , TX, 77843.*
E-mail: himgupt2@tamu.edu

Manuscript received April 19, 2018

2 PRELIMINARIES

This section reviews some basic definitions used in the paper and in general, taken mostly from [2], [7].
A secure multiparty computation is composed of probabilistic polynomial-time Turing Machines which are referred as parties in this context. Each party receives an unary input of the security parameter of the protocol 1^n. These parties perform the computation in rounds where they send and receive messages on communication channel. The number of rounds r is a function of the security parameter n.

2.1 Real vs. Ideal Paradigm

Security of multiparty computations follows the standard techniques from [8] where we consider an ideal model for the protocol to be executed. A trusted party in the ideal model captures the security requirement of the task. In the real world we can show that the adversary can not do more harm in the ideal world if the trusted party was removed and replaced by an unfair sub-protocol.

2.2 Statistical Distance

Two Random variables A and B have a statistical distance defined by the below function

$$SD(A,B) = \frac{1}{2}\sum_{\alpha} |\Pr[A=\alpha] - \Pr[B=\alpha]|$$

2.3 1/p-Indistinguishability

A distribution ensemble $X = \{X_{a,n}\}_{a\in\{0,1\}^*, n\in\mathbf{N}}$ and $Y = \{Y_{a,n}\}_{a\in\{0,1\}^*, n\in\mathbf{N}}$ are computationally $1/p$-indistinguishable, denoted by $X \overset{1/p}{\approx} Y$, if for every non-uniform polynomial time algorithm D there exists a negligible function $\mu(\cdot)$ such that for every n and every $a \in \{0,1\}^*$

$$|\Pr[D(X_{a,n})=1] - \Pr[D(Y_{a,n})=1]| \leq \frac{1}{p(n)} + \mu(n)$$

3 TWO PARTY PROTOCOL

We demonstrate in brief the main idea of our two party protocol and argue that this version provides more fairness in selecting the corrupt party. The protocol is described in two modules, the share generation and the coinflip for r rounds. The protocol also uses message authentication codes for share verification. A formal description of share generation is provided in the Algorithm 1 $Sharegen_r$.

The share generation algorithm chooses a threshold round i^* where the value of output c is revealed to both parties. This threshold round is selected uniformly over r and all the values before this round are taken from the share a_i and b_i. Every round each party shares the other half of share with other party along with the MAC tag. Parties P_1 and P_2 recreate the value a_i and b_i in every round i. When both parties complete the r round protocol they output the value a_r and b_r which is the output bit of the protocol. However, if a party aborts in a particular round i then the other party reconstructs the value from previous round and outputs that value.

Input: Security Parameter 1^n
Computation:

1) Choose $i^* \in \{1,2,3,...,r\}$ uniformly at random.
2) Define values $a_1, a_2, ..., a_r$ and $b_1, b_2, ..., b_r$ as below
   a) For $1 \leq i \leq i^*-1$ choose $a_i \leftarrow D_1, b_i \leftarrow D_2$ independently and uniformly at random.
   b) Select $c \in \{0,1\}$ uniformly at random for $i^* \leq i \leq r$, where $a_i = b_i = c$
3) Create random shares of a_i and b_i using 2-out-of-2 secret sharing scheme which gives (a_i^1, a_i^2) and (b_i^1, b_i^2)
4) Create keys and Mac tags for a_i^2 and b_i^1 where $t_i^a = Mac_k(i||a_i^2)$ and similarly for shares of b_i^1

Output:

1) Party P_1 receives the values $a_1^1, ..., a_r^1$, $(b_1^1, t_1^b), (b_r^1, t_r^b)$ and $k_a = (k_1^a, ...k_r^a)$.
2) Party P_2 receives the values $(a_1^2, t_1^a), (a_r^1, t_r^a)$, $b_1^2, ..., b_r^2$ and $k_b = (k_1^b, ...k_r^b)$.

Algorithm 1: $ShareGen_r$

Phase 1:

1) Parties P_1 and P_2 run the $ShareGen_r$ algorithm
2) If any party recieves $\perp$ from the previous algorithm, it outputs a uniformly chosen bit.
3) Each party gets the output from $ShareGen_r$ denoted by S_1 and S_2.

Phase 2: In each round $i = 1, .., r$ **do:**

1) Each party evaluates the random variable $h_i = b_{i-1}^1 \oplus a_{i-1}^2$, P_1 sends first if the value is 0.
2) if $h_i = 0$, P_1 sends the next share to P_2 first:
   a) P_1 sends (b_i^1, t_i^b) to P_2
   b) P_2 receives the input and verifies the MAC tag by running $Vrfy_{k_i^b}(i||b_i^1, t_i^b)$, if invalid P_2 outputs b_{i-1} and halts.
   c) If Mac tag is valid P_2 reconstructs b_i using the shares b_i^1 and b_i^2.
3) if $h_i = 1$, P_2 sends the next share to P_1 first:
   a) P_2 sends (a_i^2, t_i^a) to P_1
   b) P_1 receives the input and verifies the MAC tag by running $Vrfy_{k_i^a}(i||a_i^2, t_i^a)$, if invalid P_1 outputs a_{i-1} and halts.
   c) If Mac tag is valid P_1 reconstructs a_i using the shares a_i^1 and a_i^2.

Algorithm 2: $CoinFlip_r$

The next problem we encounter is that in each round a particular party always sends the share first, say P_2. Therefore P_1 always learns the output in each round first and creates a simple bias by guessing the round i^* (by probability $1/r$) and aborting in the same round if it does not likes the particular value a_i.

We present the following variation from the original protocol [2] which removes the bias of selecting a party to send the share. In each round i parties reconstruct the value $h_i = b_{i-1}^1 \oplus a_{i-1}^2$ where h_i is the random variable representing the party to send the share first in round i. Therefore in each round the probability of sending a share first is uniform. This also eliminates the choice of selecting

a particular party corrupted by adversary. The value for round 1 is uniformly chosen by the trusted party. Algorithm 2 gives the formal description of the coin flipping protocol.

3.1 Correctness Analysis

We consider the distribution of adversary and output of honest party in the real and ideal model where we follow the techniques from [8] to prove the securuty of protocol. Consider the joint distribution of Adversary **A**'s view and the output of honest party in the two models. We have the following cases:

1) **A** aborts before the round i^*: The view of the adversary is same in both model and the honest party outputs a bit which is independent of the adversary's view.
2) **A** aborts after the round i^* or does not abort: In this case also the distributions are identical. Honest party outputs the bit which is already seen by both parties.
3) **A** aborts in the round i^* and shares the value first: In this case the honest party learns the output first and will output a bit which is independent of adversary's view in both model as they both learn the final output.
4) **A** aborts in the round i^* and sees the value first: This is the only case where the adversary can bias the output. Adversary's view is identical in both models but the output of honest party is the random bit c shared by the trusted party in ideal model. In real model the p_2 outputs the bit b_{i-1} which is a random bit from D_2 and indpendent of **A**'s view. However the adversary sees this bit first by probability $\frac{1}{2}$ and selects the round correctly by $\frac{1}{r}$.

This simpy illustrated the fact that adversary only creates the bias if it is able to guess the threshold round i^* correctly. Therefore the statistical distance between the above distributions is at most $\frac{1}{2r}$

Claim 1. *There exists an efficient adversary that can corrupt party P_1^* and can bias the output of coin flipping protocol described in Algorithm 2 by $\frac{(1-2^{-r})}{4r}$*

Proof. We consider an adversary that aborts the protocol in a round $a_i = 0$. The random variable *Abort* corresponds to the round in which adversary aborts. We note here that the output of P_2 is independent of adversary's view when $h_i = 0$, i.e. the adversary ends up sending the share first in the round when it aborts. This happens with uniform proability. c_2 denotes the random variable corresponding to the output of P_2. We can say for every i the probability of output being 1 conditioned on the event that honest party sends the share first and it is the threshold round:

$$\Pr[c_2 = 1 \mid h_i = 1 \wedge i^* = i]$$

We expand this term on the condition of all the values that *Abort* can take or if it never aborts:

$$= \sum_{j=1}^{i} \Pr[c_2 = 1 \mid Abort = j \wedge h_i = 1 \wedge i^* = i].$$
$$\Pr[Abort = j \mid h_i = 1 \wedge i^* = i] +$$
$$\Pr[c_2 = 1 \mid Abort = \perp \wedge h_i = 1 \wedge i^* = i].$$
$$\Pr[Abort = \perp \mid h_i = 1 \wedge i^* = i]$$

As described above, the adversary aborts at the first round when it sees the bit $a_j = 0$, which happens when all the previous bits are 1 and last bit as 0. The event that it never aborts is when all of the bits $a_j = 0$. Therefore we have:

$$= \sum_{j=1}^{i} \Pr[c_2 = 1 \mid Abort = j \wedge h_i = 1 \wedge i^* = i].$$
$$\Pr[a_1 = ...a_{j-1} = 1, a_j = 0] +$$
$$\Pr[c_2 = 1 \mid Abort = \perp \wedge h_i = 1 \wedge i^* = i].$$
$$\Pr[a_1 = ...a_i = 1]$$

The conditional probability of output being 1 is uniform given the abort and threshold round. The output is always 1 if the adversary does not aborts. This gives,

$$= \sum_{j=1}^{i} \frac{1}{2}.\frac{1}{2^j} + 1.\frac{1}{2^i}$$
$$= \frac{\frac{1}{2}}{1-\frac{1}{2}}(1-2^{-i}).\frac{1}{2} + \frac{1}{2^i}$$

Therefore we get:

$$\Pr[c_2 = 1 \mid h_i = 1 \wedge i^* = i] = \frac{1}{2} + \frac{1}{2^{i+1}} \quad (1)$$

Now we evaluate the total probability of $c_2 = 1$, conditioned on the event which party shares the bit first

$$\Pr[c_2 = 1] = \Pr[c_2 = 1 | h_i = 0].\Pr[h_i = 0] +$$
$$\Pr[c_2 = 1 | h_i = 1].\Pr[h_i = 1]$$

The condition where the adversary sends the share first gives an uniform bit as adversary learns the output after the honest party has already seen the other share. This follows from the correctness analysis. The second term can be conditioned further according to Equation (1).

$$= \Pr[c_2 = 1 | h_i = 0] \Pr[h_i = 0] +$$
$$\sum_{i=1}^{r} \Pr[i^* = i] \Pr[c_2 = 1 \mid h_i = 1 \wedge i^* = i].\Pr[h_i = 1]$$

The probability to send the share first is also $1/2$, i.e. h_i takes value 0 or 1 uniformly. The second term is replaced from Equation (1).

$$= \frac{1}{2}.\frac{1}{2} + \sum_{i=1}^{r} \frac{1}{r}(\frac{1}{2} + \frac{1}{2^{i+1}}).\frac{1}{2}$$
$$= \frac{1}{4} + (\frac{r}{4r} + \frac{1}{4r}\sum_{i=1}^{r} \frac{1}{2^i})$$
$$= \frac{1}{2} + \frac{1}{4r}\frac{\frac{1}{2}}{1-\frac{1}{2}}.(1-2^{-r})$$
$$= \frac{1}{2} + \frac{1}{4r}(1-2^{-r})$$
$$\Pr[c_2 = 1] = \frac{1}{2} + \frac{(1-2^{-r})}{4r}$$

□

4 THE MULTIPARTY CASE

We now describe a simple extension of the above protocol to multiparty case. Consider a set of m parties in the protocol and partition them in two sets simulating Alice and Bob. The difference here is that only one party simulates Alice and remaining form a group representing Bob. The adversary can corrupt parties in any of the group but Alice represents a single party and gives full access to adversary **A**. However the majority of parties in B is honest and thus the adversary does not gets the bits in B.

The protocol proceeds in a similar fashion where a trusted party selects the shares in $2 - out - of - 2$ secret sharing scheme. The shares of party B are then shared by a Shamir's $m/2 - out - of - (m-1)$ secret sharing. The party then uniformly selects $i^* \in \{1, 2, ..., r\}$ and sets the remaining bit of A and B to the result $c \in \{0, 1\}$

This extension of the two party case is again $O(1/r)$-secure and can be seen from the argument below.

1) When p_1 is biased, the parties in p_2 have an honest majority and can reconstruct the bit b_{i-1} in case p_1 aborts. It can be seen that the adversary can not construct this bit on its own and therefore the honest party outputs a random bit.
2) If in a particular round, less than $m/2$ parties are active in p_2 then p_1 reconstructs a_{i-1} and broadcasts the same as output. As p_1 is honest, it will be a valid broadcast.
3) If atleast $m/2$ parties are active and no furhter aborts occur, the protocol proceeds as normal.

The security follows from the previous argument in two party case and is therefore $O(1/r)$-secure.

5 THREE PARTY PROTOCOL

This section describes the three party protocol following the construction from [5]. There are r rounds of interaction with several phase depending on the state. The output of protocol is selected uniformly by the trusted party and disclosed in the threshold round i^*. The construction of this protocol deviates from the previous two party model and involves a subset of parties as described in the protocol. Each of these subsets receive a uniformly selected bit σ_J^i where J is the subset group. After round i^* each of these subset bits are set to the result value and parties output the same at end of round r.

In the real model these bits will be shared in a two layer inner and outer secret sharing scheme. The adversary is assumed to be rushing and thus gets the bit for the subgroup where it has majority or exceeds the threshold. If a party aborts, the honest parties reconstruct the value of a subset in a way that adversary does not knows this value.

The protocol tolerates upto $2m/3$ malicious parties and therefore we have 2 corrupted parties in the description. Stating informally, the protocol has a bias of $O(2^3/r)$ for computing the coin toss. The proof is provided in the next section but it can be seen easily that the bias is only seen when the adversary is able to guess the round i^* and any abort before this round does not reveals anything about the output of protocol.

We now describe in brief the protocol with 3 parties in the real world setting assuming no trusted dealer. Also note that there will be an underlying signature scheme that is used to detect cheating parties and deviation from protocol but the description is omitted. The protocol is divided in two different algorithm as share generation and coin tossing.

Dividing the parties in subgroups:
$p_j \in \{p_1, p_2, p_3\}$
$Q_1 = \{p_1\}, Q_2 = \{p_2\}, Q_3 = \{p_3\}, Q_4 = \{p_1, p_2\}, Q_5 = \{p_1, p_3\}, Q_6 = \{p_2, p_3\}$
The threshold value o_J for each subset is 1 for $\{Q_1, Q_2, Q_3\}$ and 2 for $\{Q_4, Q_5, Q_6\}$
Computing default bits:

1) Choose $c \in \{0, 1\}$ and $i^* \in \{1, 2, ..., r\}$ uniformly.
2) For each round $i \in \{1, 2, ..., r\}$ and for each Q_j
   a) If $i < i^*$ select independent and uniform bits σ_J^i for each subset Q_J
   b) If $i \geq i^*$ set each $\sigma_J^i = c$, the output.

   Inner secret sharing scheme:
3) Create random secret shares of σ_J^i in $o_J - out - of - |Q_J|$ secret sharing scheme. Let $S_j^{i,J}$ be the share of party $p_j \in Q_J$.

   Outer secret sharing scheme:
4) Share each $S_j^{i,J}$ in a $3 - out - of - 3$ secret sharing scheme according to construction 1 defined in Appendix A.

Algorithm 3: $ShareGen_3$

Interaction Rounds
Preliminary phase

1) If any party aborts premature termination is executed with $i = 1$ In each round $i \in \{1, 2, ...r\}$
2) Each party p_j broadcasts its share in the outer secret sharing scheme and other parties verify the validity of the message.

   Premature Termination step
3) If $i = 1$, the remaining active parties use the two party protocol and output the resulting bit and halt.

   After last round r
4) Each party broadcasts the share of inner secret sharing, reconstruct the bit σ_J^i output it and halts.

Algorithm 4: $CoinToss_3$

5.1 Correctness Analysis

The analysis takes a different approach from the two party version. We consider two random variables (V, C) describing the view of adversary and output of honest party respectively. The other variables REAL and IDEAL describe the coin tossing protocol in real and ideal model where (V, C) are respectively defined. The motivation is to show that the statistical distance between these two variable is $O(2^3/r)$, where 3 is the number of bits(σ_J^i) to which the adversary has access. The

variables are defined as $REAL = (V_{REAL}, C_{REAL})$ and $IDEAL = (V_{IDEAL}, C_{IDEAL})$.

Claim 2. *For every non-uniform polynomial time adversary A corrupting $t < 2m/3$ parties in a coin tossing protocol described in $CoinToss_3$, the following random variables are $1/p(n) = O(2^3/r)$ close(3 is the number of bits adversary has, in general 2^{k+1}):*

$$SD(IDEAL(1^n), REAL(1^n)) \leq 1/p$$

Proof. The proof follows by showing that the adversary learns atmost 2^{k+1} bits in a round(which is 3 in this case) and can guess the round by probability $1/r$. This bounds the probability of guessing the round i^* by $O(2^3/r)$. We now see how this holds.

The first argument is to see that the statistical distance between the two variable is bounded by the probability of guessing the threshold round i.e. the round when output of the protocol is revealed. This follows from the fact that the simulator used for the ideal model selects value for σ_J^i to be sent to groups independent of the output bit. If the abort is done after the round i^*, the view of the corrupted party is same in the ideal and real model as all the bits are set to output. The only possible way to create the bias is to guess the simulation round i^* for which we bound the probability below.

The variable $\mathbf{V}$ denotes the set of all possible views of the adversary when it aborts and $\mathbf{V_i} \subseteq \mathbf{V}$ where the adversary A aborts in round i. We now evaluate the probability when threshold round $i^* = i$ and $i^* = r$. Till this round, view of the previous round happens with same proabability:

$$\Pr[V_{REAL}^{i-1} = v_{i-1} \mid i^* = i] = \Pr[V_{REAL}^{i-1} = v_{i-1} \mid i^* = r] \tag{2}$$

Now consider the view when $i^* = i$, where only two possible events are all bits being 0 or 1 each with probability $1/2$. However if $i^* > i$ then all bits being equal happens with the probability $1/2^\alpha$ where α is the number of bits the adversary has access. Now for every $v \in \mathbf{V_i}$ we can say:

$$\begin{aligned} &\Pr[V_{REAL} = v \mid V_{REAL}^{i-1} = v_{i-1} \wedge i^* = i] \\ &= \frac{2^\alpha}{2}.\Pr[V_{REAL} = v \mid V_{REAL}^{i-1} = v_{i-1} \wedge i^* = r] \end{aligned} \tag{3}$$

From Equation (2) and (3) we can write for every $v \in \mathbf{V_i}$, the probability

$$\Pr[V_{REAL} = v \mid i^* = i]$$

We expand this on the conditional probability when $V_{REAL}^{i-1} = v_{i-1}$ to get:

$$\begin{aligned} &= \Pr[V_{REAL} = v \mid V_{REAL}^{i-1} = v_{i-1} \wedge i^* = i]. \\ &\Pr[V_{REAL}^{i-1} = v_{i-1} \mid i^* = i] \end{aligned}$$

Now we replace the terms from (2) and (3) in the above equation.

$$\begin{aligned} &= 2^{\alpha-1} \Pr[V_{REAL} = v \mid V_{REAL}^{i-1} = v_{i-1} \wedge i^* = r]. \\ &\Pr[V_{REAL}^{i-1} = v_{i-1} \mid i^* = r] \end{aligned}$$

By eliminating the condition of the view in previous round we get:

$$= 2^{\alpha-1}.\Pr[V_{REAL} = v \mid i^* = r] \tag{4}$$

Now we evaluate the probability that the adversary aborts in round i^*. Consider the random variable $Abort$ which takes the value of round in which adversary A aborts. We expand this term on the conditional probability of a specific round and sum it for all r. Therefore,

$$\Pr[Abort = i^*] = \sum_{i=1}^{r} \Pr[Abort = i^* \mid i^* = i].\Pr[i^* = i]$$

Now the probability to abort in a particular round represents the set of all views the adversary has in that particular round which is that same as below:

$$= \sum_{i=1}^{r} \Pr[i^* = i] \sum_{v \in \mathbf{V_i}} \Pr[V_{REAL} = v \mid i^* = i]$$

The first tems $i^* = i$ is an uniform probability of $1/r$ and in the second term we replace it from Equation (4), which gives

$$= \sum_{i=1}^{r} \frac{1}{r}. \sum_{v \in \mathbf{V_i}} 2^{\alpha-1}.\Pr[V_{REAL} = v \mid i^* = r]$$

The sum over rounds i over the subset $\mathbf{V_i}$ simply gives us the set of all possible views the adversary has given a particular abort round. Therefore the above double sum can be written as:

$$= \frac{2^{\alpha-1}}{r} \sum_{v \in \mathbf{V}} \Pr[V_{REAL} = v \mid i^* = r] \tag{5}$$

We can take a simple bound of sum of all the probabilities in the above equation to be less than 1, i.e.

$$\sum_{v \in \mathbf{V}} \Pr[V_{REAL} = v \mid i^* = r \leq 1 \tag{6}$$

Thus giving an upper bound on $\Pr[Abort = i^*]$ as

$$\Pr[Abort = i^*] \leq \frac{2^{\alpha-1}}{r}$$

□

6 Conclusion

Section 3 and 5 presents the upper bound on the bias of coin flipping protocols in different settings. The variation in two party setting to randomize the sharing part acheives $O(1/4r)$ security without increasing the number of shares. This can be seen from the results in [2] where similar bias is acheived with $2r$ shares. The advantage of this model is that this can be extended easily to the multiparty case with extensive use of shamir's secret sharing scheme.
The three party version followed from [5] presents a similar bound of $O(2^\alpha/r)$ where specifically it is $O(2^3/r)$ (α is the number of subgroup bits over which the adversary has control).

REFERENCES

[1] *R. Cleve. Limits on the security of coin flips when half the processors are faulty. In Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC), pages 364369, 1986.*

[2] *T. Moran, M. Naor, and G. Segev. An optimally fair coin toss. In Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2009, pages 118, 2009.*

[3] *M. Blum. Coin flipping by telephone - A protocol for solving impossible problems. In Proceedings of the 25th IEEE Computer Society International Conference, pages 133-137, 1982.*

[4] *M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In Proceedings of the 20th Annual ACM Symposium on Theory of Computing, pages 1-10, 1988.*

[5] *A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with dishonest majority. In Advances in Cryptology CRYPTO '10, pages 538-557, 2010.*

[6] *I. Haitner and E. Tsfadia. An almost optimally fair three-party coin-flipping protocol. www.cs.tau.ac.il/~iftachh/papers/3PartyCF/QuasiOptimalCF_Full.pdf, 2014. Manuscript.*

[7] *S. D. Gordon and J. Katz. Partial fairness in secure two-party computation. In Henri Gilbert, editor, Advances in Cryptology EUROCRYPT 2010, volume 6110 of Lecture Notes in Computer Science, pages 157176. Springer-Verlag, 2010*

[8] *R. Canetti. Security and composition of multiparty cryptographic protocols. Journal of Cryptology, 13(1):143-202, 2000.*

[9] *N. Alon and M. Naor. Coin- ipping games immune against linear-sized coalitions. SIAM Journal on Computing, pages 46-54, 1993.*

[10] *M. Ben-Or and N. Linial. Collective coin fipping. ADVCR: Advances in Computing Research, 5, 1989.*

[11] *U. Feige. Noncryptographic selection protocols. In Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS), 1999.*

[12] *A. Ambainis. A new protocol and lower bounds for quantum coin fipping. J. Comput. Syst. Sci., 68(2):398-416, 2004.*

[13] *A. Ambainis, H. Buhrman, Y. Dodis, and H. Rohrig. Multiparty quantum coin ipping. In Proceedings of the 18th Annual IEEE Conference on Computational Complexity, pages 250- 259, 2004.*

[14] *R. Cohen, Y. Lindell. Fairness versus guaranteed output delivery in secure multiparty computation, in Advances in CryptologyASIACRYPT 2014, part II. (2014) pp. 466485*

APPENDIX A
CONSTRUCTION 1

$\alpha - $ out $-$ of $-$ m secret sharing:

Let s be a secret to be shared among m parties with respect to party p_j. The share is divided by:

1) Create to shares (s^1, s^2) in $2 - out - of - 2$ secret sharing scheme. These are the mask and complemen shares denoted by $mask(s)$ and $comp(s)$.
2) Create shares $\lambda^1, ...\lambda^{\alpha-1}$ of $comp(s)$ in $(\alpha - 1) - out - of - m$ secret sharing scheme.

Party p_j gets the share $mask(s)$ and rest of the parties get the complement shares λ^j generated in step 2 above.