Limits on the Security of Coin Flips When Half the Processors are Faulty

(Extended Abstract)

Richard Cleve
Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4

1. Introduction

Protocols which allow an asynchronous network of processors to agree on a random (unbiased) bit are proposed in [1] and [4]. It is claimed that (assuming a trapdoor function exists), if less than half of the processors are faulty then the correct processors will still agree on a bit whose bias is negligibly small (when the running time of the processors is $poly(n)$ the bias is smaller than $O(\frac{1}{n^k})$ for all k). If half the processors are faulty then these protocols are no longer effective: the bits output by the correct processors may be heavily biased.

We prove that the above protocols are optimal in the sense that no protocol exists which tolerates faults in at least half of the processors. The result is very general because few restrictions are made on the types of communication allowed between correct processors (such as private channels and global channels) and the correct processors only need to agree on a bit in a weak probabilistic sense. Also, the faulty processors do not require very much power. They can privately communicate with each other but they cannot read messages which are exchanged privately between two correct processors.

An interesting instance of the problem arises when the number of processors is fixed at two and one of them may be faulty. This is the so-called coin flipping by telephone problem which is proposed in [3] and cannot be solved with very high security. There are protocols for 2-processor coin tossing which (assuming that a trapdoor function exists) achieve a weaker level of security (these are discussed in section 4). The processors run in $poly(n)$ time and the bias of the bit which a correct processor outputs is less than $O(\frac{1}{n^k})$ for some fixed k. More precisely, the bias will be less than $O(\frac{1}{\sqrt{r}})$ where r is the number of rounds of communication in the protocol. For many applications (such as secret exchanging [5]) this weaker level of security is sufficient. In section 2 it is proven that no 2-processor protocol exists for which the bias of the output of a correct processor is less than $O(\frac{1}{r})$.

In [4] it is pointed out that multiprocessor coin tossing schemes have their application in the problem of randomly choosing a leader in a network of processors and the problem of fairly allocating resources within a network.

2. 2-Processor Coin Tossing Schemes

In 2.1, 2-Processor coin tossing schemes are defined precisely and, in 2.2, a lower bound on the security of 2-processor coin tossing schemes is proven.

2.1 Definitions

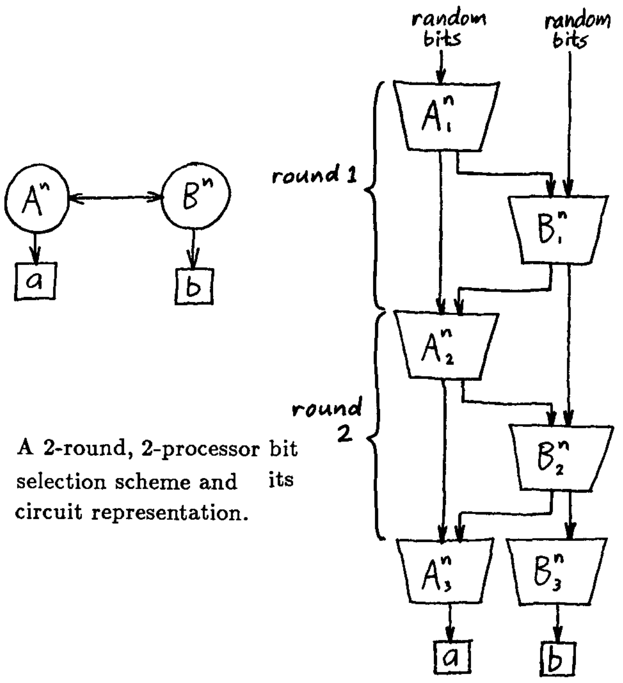
A ***2-processor bit selection scheme*** is a sequence of pairs of processors $\{(A^n, B^n)\}_{n=1}^{\infty}$ with the following properties. For each n, A^n and B^n each have access to a private supply of random bits and they can communicate with each other. If the system is executed then A^n, B^n will output bits a, b (respectively) within $poly(n)$ time.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1986 ACM 0-89791-193-8/86/0500/0364 $00.75

Without any loss of generality, it can be assumed that the operation of the system consists of $r(n)$ *rounds* (for some $poly(n)$ bounded function r), where a *round* consists of the following events. A^n performs some computations (in $poly(n)$ time) and then sends a message to B^n and then B^n performs some computations (in $poly(n)$ time) and sends a message to A^n.

More formally, a processor can be viewed as a sequence of $r(n) + 1$ circuits, each of which is $poly(n)$ bounded in size. The first $r(n)$ circuits simulate the bevavour of the processor at the $r(n)$ rounds and the last circuit outputs the bit which the processor selects. Outputs of the first $r(n)$ circuits are state information and messages to the other processor. Inputs to the circuits are random bits, state information from previous rounds or messages from the other processor.



A 2-round, 2-processor bit selection scheme and its circuit representation.

For a 2-processor bit selection scheme to qualify as a coin tossing scheme, the bits that the two processors select should be in agreement and they should be random.

A strong definition of agreement would be the condition that, when the bit selection scheme is run, $a = b$. It may be sufficient for some purposes to adopt a weaker definition of agreement such as $\Pr[a = b] \geq 1 - O(\frac{1}{n^k})$ for all k. Define a 2-processor bit selection scheme to be *ϵ-consistent* (where $\epsilon > 0$) if $\Pr[a = b] \geq \frac{1}{2} + \epsilon$. Note that ϵ-consistency is a considerably weaker property than the two definitions of agreement proposed above.

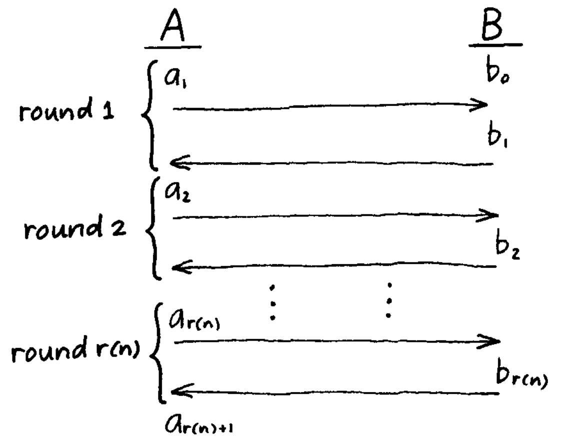
Randomness could be defined as the property that a and b have a *bias* of zero, where the *bias* of a bit x is defined as $|\Pr[x = 0] - \frac{1}{2}|$ (and the *bias towards 0* of bit x is defined as $\Pr[x = 0] - \frac{1}{2}$ (the *bias towards 1* is defined similarly)). Again, it may suffice to have a weaker definition of randomness such as the condition that the bias of both a and b be bounded by $O(\frac{1}{n^k})$ for all k. The definition of security, which is given below, implies this second randomness condition.

Clearly, if one of the two processors is faulty then it is unrealistic to expect agreement since the faulty processor could output a bit which the correct processor knows nothing about. It is desirable, however, for the output of the correct processor to still be random. Define a 2-processor bit selection scheme $\{(A^n, B^n)\}_{n=1}^{\infty}$ to be *secure* if, for any polynomial p, there exists a function f, with $f(n) \leq O(\frac{1}{n^k})$ for all k such that the following holds. For all n, if one of A^n, B^n is replaced by a faulty processor which runs in time $p(n)$ then the bias of the output of the other processor is less than $f(n)$. Thus, as n gets large, no faulty processor of feasable running time can inflict a significant bias on the output of the other processor. Note that security is defined in such a way that the security condition implies the second randomness condition proposed in the previous paragraph.

2.2 Impossibility Result

Theorem: If $\{(A^n, B^n)\}_{n=1}^{\infty}$ is an ϵ-consistent, 2-processor bit selection scheme then $\{(A^n, B^n)\}_{n=1}^{\infty}$ is not secure. In fact, for each n, there exists a faulty processor which causes the bias of the output of the correct processor to be at least $\frac{\epsilon}{4r(n)+1}$, where $r(n)$ is the number of rounds of communication of the scheme.

Proof: We say that a processor *quits* at round i if, from round i onwards, all the messages that the processor sends are strings of zeroes. If, say, B^n quits at some round i in the middle of the protocol then A^n will still output some bit $\tilde{a}$ which we shall refer to as a *default* output. Note that A^n could always compute and store $\tilde{a}$ at the beginning of round i before receiving any messages from B^n in round i (but A^n cannot, in general, compute $\tilde{a}$ before round i). For $i \in \{1, 2, ..., r(n)\}$, let a_i be the default output of A^n if B^n quits at round i and let $a_{r(n)+1}$ be the output of A^n if B^n does not quit during the protocol. Also, for $i \in \{0, 1, ..., r(n) - 1\}$, let b_i be the default output of B^n if A^n quits the protocol at round $i+1$ and let $b_{r(n)}$ be the output of B^n if A^n does not quit during the protocol. Note that A^n and B^n can compute bits a_i and b_i (respectively) before they send their message in round i.



The default outputs are marked at the point in the protocol where they can be computed.

We now define $4r(n) + 1$ faulty processors: $\tilde{A}^n$, A^n_{10}, $A^n_{20}, ..., A^n_{r(n)0}, A^n_{11}, A^n_{21}, ..., A^n_{r(n)1}, B^n_{10}, B^n_{20}, ..., B^n_{r(n)0}, B^n_{11}, B^n_{21}, ..., B^n_{r(n)1}$. We shall eventually show that at least one of these processors biases the output of the correct processor by at least $\frac{\epsilon}{4r(n)+1}$. Faulty processor $\tilde{A}^n$ is very simple. $\tilde{A}^n$ always quits at round 1. The output of B^n when $(\tilde{A}^n, B^n)$ is run will be b_0 so $\tilde{A}^n$ biases the output of B^n by

$$\max\{\Pr[b_0 = 0], \Pr[b_0 = 1]\} - \frac{1}{2}.$$

For each $i \in \{1, 2, ..., r(n)\}$, faulty processors $A^n_{i0}, A^n_{i1}, B^n_{i0}$, B^n_{i1} operate as follows.

```
$A^n_{i0}$: simulate $A^n$ for rounds $1, 2, ..., i-1$
      compute $a_i$
      if $a_i = 0$ then
         simulate $A^n$ for round $i$
         quit at round $i+1$
      else
         quit at round $i$

$A^n_{i1}$: simulate $A^n$ for rounds $1, 2, ..., i-1$
      compute $a_i$
      if $a_i = 1$ then
         simulate $A^n$ for round $i$
         quit at round $i+1$
      else
         quit at round $i$

$B^n_{i0}$: simulate $B^n$ for rounds $1, 2, ..., i-1$
      compute $b_i$
      if $b_i = 0$ then
         simulate $B^n$ for round $i$
         quit at round $i+1$
      else
         quit at round $i$

$B^n_{i1}$: simulate $B^n$ for rounds $1, 2, ..., i-1$
      compute $b_i$
      if $b_i = 1$ then
         simulate $B^n$ for round $i$
         quit at round $i+1$
      else
         quit at round $i$
```

Since A^n and B^n run in $poly(n)$ time, so do all these faulty processors.

The bias towards 0 of the output of B^n when (A^n_{i0}, B^n) is run is

$$\Pr[a_i = 0 \wedge b_i = 0] + \Pr[a_i = 1 \wedge b_{i-1} = 0] - \frac{1}{2}.$$

The bias towards 1 of the output of B^n when (A^n_{i1}, B^n) is run is

$$\Pr[a_i = 1 \wedge b_i = 1] + \Pr[a_i = 0 \wedge b_{i-1} = 1] - \frac{1}{2}.$$

The bias towards 0 of the output of A^n when (A^n, B^n_{i0}) is run is

$$\Pr[b_i = 0 \wedge a_{i+1} = 0] + \Pr[b_i = 1 \wedge a_i = 0] - \frac{1}{2}.$$

The bias towards 1 of the output of A^n when (A^n, B^n_{i1}) is run is

$$\Pr[b_i = 1 \wedge a_{i+1} = 1] + \Pr[b_i = 0 \wedge a_i = 1] - \frac{1}{2}.$$

Let Δ be the average of these $4r(n)+1$ biases. Then

$$\Delta = \frac{1}{4r(n)+1}\Big[\max\{\Pr[b_0 = 0], \Pr[b_0 = 1]\} - \frac{1}{2}$$

$$+\sum_{i=1}^{r(n)}(\Pr[a_i = 0 \wedge b_i = 0] + \Pr[a_i = 1 \wedge b_{i-1} = 0] - \frac{1}{2}$$

$$+\Pr[a_i = 1 \wedge b_i = 1] + \Pr[a_i = 0 \wedge b_{i-1} = 1] - \frac{1}{2}$$

$$+\Pr[b_i = 0 \wedge a_{i+1} = 0] + \Pr[b_i = 1 \wedge a_i = 0] - \frac{1}{2}$$

$$+\Pr[b_i = 1 \wedge a_{i+1} = 1] + \Pr[b_i = 0 \wedge a_i = 1] - \frac{1}{2})\Big].$$

The above equation reduces to

$$\Delta \geq \frac{\epsilon}{4r(n)+1}$$

because

$$\Pr[a_i = 0 \wedge b_i = 0] + \Pr[a_i = 0 \wedge b_i = 1] + \Pr[a_i = 1 \wedge b_i = 0]$$

$$+\Pr[a_i = 1 \wedge b_i = 1] = 1$$

for $i \in \{1, 2, ..., r(n)-1\}$ and

$$\Pr[a_{r(n)+1} = b_{r(n)}] \geq \frac{1}{2} + \epsilon$$

and

$$\Pr[a_1 = b_0] = \Pr[a_1 = 0]\Pr[b_0 = 0] + \Pr[a_1 = 1]\Pr[b_0 = 1]$$

$$\leq \max\{\Pr[b_0 = 0], \Pr[b_0 = 1]\}.$$

The second equation follows from the fact that the scheme is ϵ-consistent and the third equation follows from the fact that a_1 and b_0 can be computed before any information is exchanged between A^n and B^n so they must be independent random variables.

Since the average of the $4r(n)+1$ biases is $\frac{\epsilon}{4r(n)+1}$, at least one of them must be greater than or equal to $\frac{\epsilon}{4r(n)+1}$ so the theorem is proved.

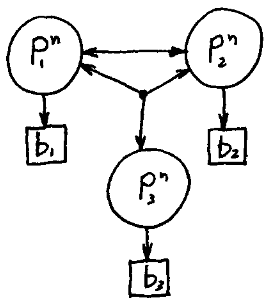
3. Multiprocessor Coin Tossing Schemes

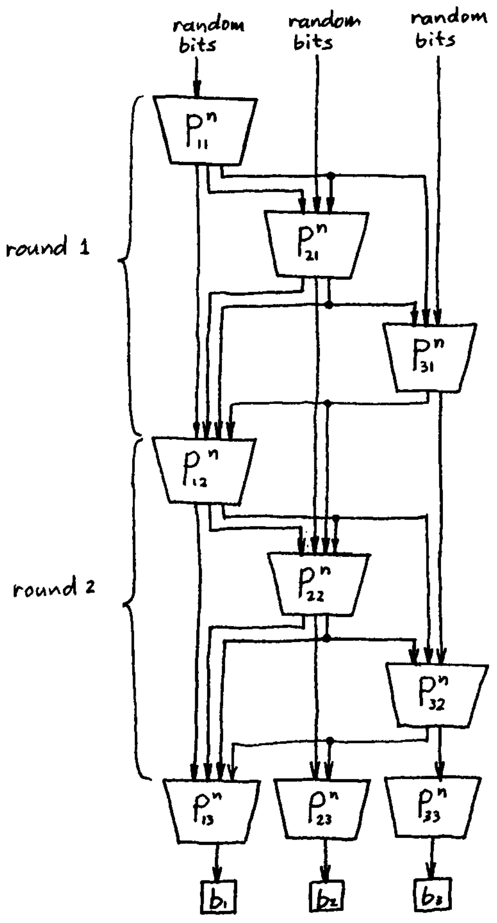
In 3.1, a generalization of a 2-processor bit selection scheme to a bit selection scheme which contains an arbitrary number of processors is made. In 3.2, it is proven that a multiprocessor coin tossing scheme cannot be very secure if half its processors are faulty.

3.1 Definitions

Let s be a polynomially bounded function (Some interesting cases are when $s(n)$ is constant or $s(n) = n$). Define an *s-processor bit selection scheme* as a sequence of tuples of processors of the form $\{(P^n_1, P^n_2, ..., P^n_{s(n)})\}_{n=1}^{\infty}$ with the following properties. For each n, P^n_1, P^n_2, ..., $P^n_{s(n)}$ are processors which have access to a private supply of random bits and which can communicate with each other. Also, each processor, P^n_i, will output a bit, b_i, within *poly*(n) time, when the system is executed. The communication between the processors is of a very general form. The network contains $c(n)$ channels, where c is a *poly*(n) bounded function. Each channel is shared by some subset of the $s(n)$ processors. Thus, the network could consist only of one global communication channel or it could consist of a channel for each pair of processors (in other words it could support only private communication). Without any loss of generality, the system operates in rounds as follows. P^n_1 performs some computations (in *poly*(n) time) and then sends a message to each channel which it has access to. Then P^n_2, P^n_3, ..., $P^n_{s(n)}$ each in turn do the same thing.

As in the case of 2-processor bit selection schemes, each processor can be formalized as a sequence of $r(n)+1$ circuits which are *poly*(n) bounded in size. These circuits simulate the behaviour of the processor at different rounds.





A 2-round, 3-processor bit selection scheme with 2 communication channels and its circuit representation.

An s-processor bit selection scheme is *ϵ-consistent* if, for all n, $\Pr[b_i = b_j] \geq \frac{1}{2} + \epsilon$ for all $1 \leq i,j \leq s(n)$.

Let t be a *$poly(n)$* bounded function. An s-processor bit selection scheme is *t-secure* if for any polynomial, p, there exists a function f, with $f(n) \leq O(\frac{1}{n^k})$ for all k such that the following holds. For all n, if not more than $t(n)$ of the processors P^n_1, $P^n_2, \ldots,$ $P^n_{s(n)}$ are replaced by faulty processors whose running time is bounded by $p(n)$ then the bias of the output of all correct processors is less than $f(n)$. The faulty processors are given a private communication channel (if one does not already exist for the processors that they replace) but the communication system is not altered in any other way. Faulty processors cannot read messages transmitted on channels which they do not belong to.

3.2 Impossibiltiy Result

Theorem: If $\{(P^n_1, P^n_2, \ldots, P^n_{s(n)})\}^{\infty}_{n=1}$ is an ϵ-consistent, s-processor bit selection scheme then it is not $\lceil \frac{s}{2} \rceil$-secure. That is, half the processors, if faulty, can bias the output of one of the other processors significantly. Specifically, for each n, there exists a set of $\lceil \frac{s(n)}{2} \rceil$ or $\lfloor \frac{s(n)}{2} \rfloor$ processors which, if faulty, can bias the output of one of the correct processors by at least $\frac{\epsilon}{4s(n)r(n)c(n)+1}$, where $r(n)$ is the number of rounds of communication and $c(n)$ is the number of communication channels in the network.

Proof: For each n, partition the $s(n)$ processors into two sets, one containing $\lceil \frac{s(n)}{2} \rceil$ processors and the other containing $\lfloor \frac{s(n)}{2} \rfloor$ processors. It is possible to simulate the operation of this scheme by two processors, A^n and B^n (which each run in *$poly(n)$* time) where A^n simulates all the processors in the first set and B^n simulates all the processors in the second set. A^n and B^n communicate with each other to simulate the exchanges of messages between processors in different sets. The number of rounds of communication that occur between A^n and B^n during the simulation is bounded by $r(n)s(n)c(n)$. Therefore, by the theorem in section 2.2, for each n, there exists either a faulty A^n or B^n which can bias the output of one of the bits of the other processor by at least $\frac{\epsilon}{4r(n)s(n)c(n)+1}$. The algorithm of the faulty processor can always be implemented within the processors that it simulates if these processors have a private communication channel. This completes the proof of the theorem.

4. Some Positive Results

If a trapdoor function, F, exists then it is possible to construct an r-round, 2-processor bit selection scheme, $\{(A^n, B^n)\}^{\infty}_{n=1}$, for which all correct processors always output the same bit, which satisfies the following weak security condition. For any polynomial p, there exists $w > 0$ such that if, for some n, one of the processors A^n, B^n is replaced by a faulty one which runs in time $p(n)$ then the bias of the bit output by the other processor is bounded by $\frac{w}{\sqrt{r(n)}}$.

$\{(A^n, B^n)\}_{n=1}^{\infty}$ operates as follows. B^n initially generates $r(n)$ random public/private key pairs (K_1, T_1), $(K_2, T_2), \ldots, (K_{r(n)}, T_{r(n)})$ and selects $r(n)$ random bits x_1, $x_2, \ldots, x_{r(n)}$ and sends the public keys K_1, $K_2, \ldots$, $K_{r(n)}$ and the encryptions $F_{K_1}(x_1)$, $F_{K_2}(x_2), \ldots$, $F_{K_{r(n)}}(x_{r(n)})$ to A^n. During round i, A^n randomly chooses a bit y_i and sends it to B^n and then B^n sends the private key T_i to A^n. Finally, A^n and B^n each output the majority of $(x_1 \oplus y_1, x_2 \oplus y_2, \ldots, x_{r(n)} \oplus y_{r(n)})$. If at some round, i, in the protocol B^n does not send the valid private key T_i which corresponds to K_i (which A^n can verify) then A^n continues the protocol substituting random bits for $x_i, x_{i+1}, \ldots, x_{r(n)}$.

We now investigate the security of $\{(A^n, B^n)\}_{n=1}^{\infty}$. If A^n is replaced by a faulty processor then the output of B^n will have very little bias since the faulty processor can never guess the value of $x_i \oplus y_i$ very well during round i (the fact that F is a trapdoor function implies this). If B^n is replaced by a faulty processor then all that B^n can do is not send T_i to A^n at some round i. This strategy can change the balance of 0s and 1s in $(x_1 \oplus y_1, x_2 \oplus y_2, \ldots, x_{r(n)} \oplus y_{r(n)})$ by at most 1 and therefore will only change the output of A^n with probability $O(\frac{1}{\sqrt{r(n)}})$.

In [1] a description of a bit selection scheme which is similar to the one described above, except that it applies to networks which contain an arbitrary number of processors, is given.

5. Open Problems

The scheme described in section 4 only has security $O(\frac{1}{\sqrt{r}})$ (this is the bound of the bias that a faulty processor can cause) while the the lower bound on security proven in section 2 is $O(\frac{1}{r})$. It would be interesting to tighten this gap.

It would also be interesting to see how much bias a group of isolated processors (i.e. which have no special facility to privately communicate with each other) could cause.

Acknowledgements

I am grateful to Charles Rackoff for introducing me to the subject of distributed coin tossing and for his many helpful remarks concerning this research.

References

[1] Awerbuch, B., Blum, M., Chor, B., Goldwasser, S., Micali, S., *How to Implement Bracha's O(log n) Byzantine Agreement Algorithm*, Manuscript, 1985.

[2] Ben-Or, M., Linial, N., *Collective Coin Flipping, Robust Voting Schemes and Minima of Banzhaf Values*, 26th FOCS, 1985.

[3] Blum, M., *Coin Flipping by Telephone-a Protocol for Solving Impossible Problems*, Spring COMPCON Conference, 1982.

[4] Broder, A., Dolev, D., *Flipping Coins in Many Pockets*, 25th FOCS, 1984.

[5] Luby, M., Micali, S., Rackoff, C., *How to Simultaneously Exchange a Secret Bit by Flipping a Symetrically Biased Coin* 24th FOCS, 1983.