Collective Coin Tossing Without Assumptions nor Broadcasting

Silvio Micali Tal Rabin

1 Introduction

To obtain security, one needs to utilize many resources. Among these are one-way functions, physically secure communication channels, and —though less well known— *broadcasting.*

We will argue, though, that this resource should not be taken for granted in a cryptographic scenario, and that actually should be removed. We will show that this can be done thanks to some recent developments in the field of distributed computation and actually hope to generate more awareness about this field for our cryptographic work.

We focus on one such a primitive, collective coin flipping. Here a group of players, some of which are dishonest, want to select a common, random and totally unbiased bit. Our desire of having the coin totally unbiased obliges us to dispens with cryptography, since else one would always have a miniscule chance of guessing the relevant secret key and bias the coin. To explain how to get perfect common coins, we need to revisit another protocol: verifiable secret sharing.

As we shall see, along the way, we will provide a very simple proof of a beautiful and unpublished, VSS protocol.

1.1 Verifiable Secret Sharing

Awerbuch, Chor, Goldwasser, and Micali [B.Mi86] introduced and cryptographically implemented the somewhat paradoxical notion of a verifiable

A.J. Menezes and S.A. Vanstone (Eds.): Advances in Cryptology - CRYPTO '90, LNCS 537, pp. 253-266, 1991.

secret sharing. This is a protocol involving a distinguished party, called the *dealer*, and additional parties, called the *players*. Any of these parties (including the dealer) may be malicious, and deviate from their prescribed instructions in an arbitrary way. Informally, the protocol consists of two stages. In the first stage the dealer "secretly commits" to a value of its choice. In the second stage this value is recovered. The value is *secret*, at the end of stage 1, in the sense that no subset of players of suitably small size can guess it better than at random, even if they exchange all the information in their possession so far (which good players never do in the first stage.) The value is *committed*, in stage 1, in the sense that a good player can VERIFY that there exists a unique value x such that whenever stage 2 is performed, with or without the help of the dealer, and *no matter what the bad players might do*, all the good players will recover x. Moreover this unique, but unknown, value x is what a good dealer chose it to be.

Applications. Verifiable secret sharing is extremely useful. It is the crucial subroutine of all recent completeness theorems for protocols with honest majority, most notably [GoMiWi86, GaHaYu87, BeGoWi88, ChCrDa88, RaBe89, BeMiRo90].

Below let us point out just two applications that will help us to illustrate some future points.

1. *Delayed Disclosure.* Assume that the president of the United States wants his generals to know the secret password for the Country's nuclear defense system, if he is killed. Then he may execute stage 1 of a VSS protocol when he is still alive and order his generals to execute stage 2 only if he gets killed. As he trusts the majority of his generals to follow his orders, the password will remain secret until he is alive. Should he get killed, again because the majority of his generals will participate in stage 2, the password is guaranteed to be recovered.

2. *Collective Coin Flipping.* Assume n parties want to agree on a common, random bit. Then each party secretly and randomly selects a bit, and commits herself to it using stage 1 of a VSS protocol. Once all have done it, stage two is executed, all the committed bits are recovered and broadcasted, and the common coin is assumed to be the sum modulo 2 of all decommitted bits.

In the first application, the emphasis is on an honest dealer who does not trust any single player, or even any minority of the players; in the second one, on an honest group of players who do not trust the dealer.

1.2 Verifiable vs Simple Secret Sharing

Secret Sharing. The earlier notion of a secret sharing was independently introduced by Blakley [Bla79] and Shamir [Sha79]. In a secret sharing protocol with parameters n and t, a *dealer* D possesses a secret string s. From s the dealer computes n other strings $s_1, ..., s_n$ —called *shares*— such that s is unpredictable given any $\leq t$ of the shares, but s is easily computable given any $t+1$ of the shares.

For instance, in [Sha79] the dealer randomly chooses a polynomial P of degree t with coefficients in Z_p, p prime, such that $P(0) = s$, and gives to player i the string $P(i)$ as his share.

The limitations of Secret Sharing. Secret sharing does not achieve secret commitment as discussed above. For instance,

- In the case of application 1, some generals may be trahitors and, during the reconstruction of the password, may contribute wrong strings as their shares of the secret. This will avoid that the good general reconstruct the password! Though the password is correctably reconstructed when given $t+1$ good shares and nothin else, all bets are off when, together with good shares, one is given also bad ones. For instance, in the implementation of [Sha79], $t+1$ shares uniquely identify a polynomial. Thus if t shares are bad out of —say— $3t+1$, as soon one chooses t shares, almost surely a bad one would be included and a wrong secret reconstructed. Cycling through all possible subsets of shares of size $t+1$ (so to identify the $2t+1$ shares that define the same polynomial) would be impossible since it would require exponential time.

- In case of application 2, every player acts as a dealer. One such dealer, D, may co-operate with the bad players as follows. He gives them good shares to begin with, but during reconstruction he tells them whether they should contribute the good shares or random ones. This allows D to bias the common coin. Assume that he wants the common coin to come up 0. If the sum of all previously decommitted bits and his own bit happens to be 0, he tells the bad players to contribute the original share he gave them. This way the reconstruction of his secret will proceed smoothly, and the common coin will be 0 as he wanted. If the sum modulo 2 of all bits is 1, he tells the bad players to contribute random strings during the reconstruction of his own bit. Thus the good players cannot reconstruct any bit for D. Now they are in trouble no matter what they do. If their strategy is to take the sum modulo 2 of the reconstructed bits, they would easily allow D to bias the coin. In fact D has two chances of having 0: one if the sum of all bits modulo 2 is 0, the other if the sum of all the bits except his own is 0. If their strategy consists of starting again the protocol without D, again D would have two chances of obtaining 0. (This interesting phenomenon was first observed by Broder and Dolev [BrDo84] in the case of 2-people coin flipping.)

Problem 1 could be solved by having the dealer digitally sign the shares he hands out, but problem 2 is of no easy solution. We need verifiable secret sharing.

1.3 Our Solution

Our protocol is the first one to simultaneously enjoy many attractive properties:

1. It cannot be defeated even if 1/3 of the players are malicious and cooperate with each other to disrupt it.

2. It works without one-way functions if the players communicate via safe lines. Thus even if the bad players have infinite computing resources, they cannot defeat it.

3. It has 0 probability of error. Thus there is *absolute certainty* that the dealer's input is committed, is secret, and will later be correctly recovered.

4. It works in constant number of rounds.

5. It works *without broadcasting.* It is enough thatevery pair of players can exchange messages.

Our protocol is in fact the first VSS protocol implementable without broadcasting. Properties 1, 2 alone were achieved by Chaum, Crépeau, and Damgård [ChCrDa88]; properties 1,2 and 3 were —independently— achieved by Ben-Or, Goldwasser, and Wigderson [BeGoWi88]; properties 1, 2, 3, and 4 were later achieved by Feldman in his Ph.D Thesis. His method has not been separately published, but appears without proof in [BeGoWi88], whose authors have also announced to have found an equivalent but much more complicated method. In this paper we also provide a simpler proof of Feldman's protocol.

Let us also say that Cynthia Dwork has told us that she, Dolev, Naor, and Yung have obtained this same result, and that they will not write it up since we have already done it.

Dispensing with Broadcasting. Removing broadcasting is useful, not only from a theoretical point of view —where one wants to know what are the resources necessary to guarantee security— but also from a practical one.

Capability of broadcasting may be obtained among processors imbedded in a special parallel machine. However, the players of cryptographic protocols are physically far away, and do not belong to a special computer hardware. In this scenario, it is hard to believe that they players may want or always can to communicate —say— via radio using established frequencies! Even if they could, this would hardly be considered a secure communication channel, unless cryptography is used. It is a main point here not to use cryptography to understand what security can still be achieved if one-way functions do not exist!

Broadcasting a message may also be simulated by sending the same message to all other players. This, however, only works if all players are honest. In a cryptographic setting, one better not assume that a player really sends the same message to all other players!

We remove broadcastas follows. First we blend Feldman's protocol with graded broadcast (a protocol due to [FeMi90]) thus obtaining a protocol that is implementable in a point-to-point communication network but still presents a degree of ambiguity. This ambiguity is then removed by running the expected constant-round Byzantine agreement of [FeMi90] a constant number of times. In this protocol, however, the players do not terminate simultaneously, but they can be off of one round. Thus some good players may start executing other steps of our protocol ahead of bad players, and, in doing so, they may divulge information before before the bad ones have sent theirs to anyone. Though this is potentially dangerous it will not affect the correctness of our enterprise. This extends also to the point when VSS is used to secretly committing to bits that are later Exclusive-Ored to obtain a common coin. Generating a common coin thus entails running concurrently n Byzantine agreements. Though each one ends in expected constant-rounds, one cannot "expect" that all of them will end in expected constant-rounds! (To see this, let's change game. Assume that an individual flips a fair coin. Then it will get Heads in expected two trials. Assume now that n people flip each a fair coin. Then it is not true that all will get Heads in expected two trials. Rather After one round half will have gotten Heads, in one more round another half will get Heads, and so $\log n$ round will be needed.) Thus we must make use of a recent result of Ben-Or and El-Yaniv [BeEl88] that extends the work of [FeMi90] to many concurrent Byzantine agreements. Though the players may even more seriously be out-of-step, one can argue that security of the coin flip is not affected.

Optimality. In the final paper, we will prove our result to be optimal in all mentioned accounts.

More on Private Channels. Protocols whose security relies on private channels are much preferable to the ones relying on one-way functions. First, because all communication can still be encrypted though private channels are available (thus an enemy must both —say– factor, and have access to physically protected lines (or human courriers). Second, because one-way functions may not exist, but secure channels may. And third, because even if an enemy manages to dig —say— a hole in the ground and tap a channel that was believed to be secure, we can consider this equivalent to having the enemy corrupting the player who owned the channel. Since we can tolerate 1/3 of the players to be corrupted, we may essentially tolerate an enemy to tap quite a few channels without compromising our security.

2 Definitions

Graded Protocol - Let P be a distributed terminating protocol, executed by n players. There is a distinguished player, D, the dealer, who starts with a private value $s \in [0..m-1]$. The protocol P is intended to distribute the value

s to the n players. At the end of the protocol each player P_i outputs a *grade*, $P\text{–}confidence_i \in \{0,1,2\}$, and is able to "access" a $P - value_i \in [0..m-1]$. The meaning of "access" depends on the type of the protocol, P. We say that P is a *graded protocol* if the following properties hold:

1. *Acceptance of good values* - If the dealer D is honest then for each honest player P_i, $P\text{–}confidence_i = 2$.
2. *Semi-unanimity* - If any honest player P_i outputs $P\text{–}confidence_i = 2$, then $P\text{–}confidence_j > 0$ for each honest player P_j.
3. *Verifiability* - There exists a value $s' \in [0..m-1]$, such that all good players whose $P\text{–}confidence_i > 0$, can access s'. If D is honest then $s = s'$.

In this paper we shall need the following definitions of two particular graded-protocols.,

Gradecast Protocol - Is a graded protocol. The meaning of "access" in this protocol is that player P_i actually holds $GCST\text{–}value_i$ at the end of the Gradecast.

Graded Share/Verify and Recover Protocol - Graded Share/Verify and Recover is a graded protocol. Player P_i "accesses" $GSV\text{–}value_i$ by executing the Recover Protocol, of which the output is $GSV\text{–}value_i$. An additional property is required for the Graded Share/Verify:

Unpredictability - Let A be an adversary acting on the protocol, who doesn't corrupt the dealer, and who can corrupt up to $t < n/3$ of the players. If A outputs a value τ as his prediction of the dealer's value before the start of *Recover*, then the probability that $\tau = s$ is $1/m$.

Verifiable Secret Sharing Protocol - Is a distributed, two phase, terminating protocol, executed by n players, and a distinguished player D, the dealer. The dealer holds a private input $s \in [0..m-1]$ which he distributes in some manner in the first phase of the protocol. At the end of the first phase the dealer will either be disqualified, or it will be known that in the second phase, the value s will be known to all honest players.

Interactive Consistency Protocol [PSL]: Is a distributed protocol carried out by n players. Player P_i has a private value v_i. The protocol allows each player to compute a vector $vector_i = b_{i1}b_{i2}...b_{in}$, so that for each honest P_i and P_j we have $vector_i = vector_j$. And for all honest players P_i and P_j we have that $vector_i[j] = v_j$. In different words, this is n Byzantine Agreements executed concurrently.

3 Graded Share/Verify and Recover

Theorem 1 *Graded Share/Verify and Recover with the above properties can be achieved in constant time, where $t < n/3$, without the use of broadcast channels and with no probability of error.*

We shall start by stating our protocol:
Graded Share/Verify Protocol

1. Dealer randomly chooses $a_{ij} \in Z_p \;\; 0 \leq i,j \leq t$ where $p > n, m$, except $a_{00} = s$, and defines a bivariate polynomial $f(x,y) = \sum_{i,j} a_{ij}x^iy^j$, so that $f(0,0) = s$. He computes $f(i,y)$ and $f(x,i)$ for all i and defines: $g_i(y) = f(i,y)$, $h_i(x) = f(x,i)$. He hands over to player P_i, on the private channel, the polynomials $g_i(y)$ and $h_i(x)$.

2. Player P_i computes $h_i(j)$ for each j and hands the value to player P_j.

3. Player P_i looks at all the values he received in the previous step, $h_1(i), ..., h_n(i)$ (some may have not been received), and checks whether they satisfy
$$g_i(j) = h_j(i)?$$
For every j that doesn't satisfy the equation, P_i gradecasts "expose $g_i(j)$".

4. The dealer gradecasts the values $g_i(j)$ for all requests that he received with $GCST{-}confidence_D = 1$ or 2.

5. Player P_i checks for all the values $g_i(j)$ and $g_k(i)$ that were gradecasted by the dealer whether:
   - $GCST{-}confidence_i = 2$.
   - Does the $GCST{-}value_i$, gradecasted by the dealer equal the value which he holds.

If either one is not satisfied he gradecasts "expose $g_i(y)$ and $h_i(x)$" and distributes on the private channelles "disqualify dealer".

6. The dealer gradecasts all the polynomials requested in the previous step for requests with $GCST\text{--}confidence_D = 1$ or 2.

7. Player P_i checks for all requests,in Step 5, whose $GCST\text{--}confidence_i = 2$: if the reply in Step 6 doesn't have $GCST\text{--}confidence_i = 2$, or for some gradecasted $g_k(y)$ and $h_k(x)$, $g_k(i) \neq h_i(k)$ or $h_k(i) \neq g_i(k)$ then he distributes "disqualify dealer".

8. Player P_i counts how many "disqualify dealer" votes he got if $count \geq t+1$ then he distributes "no secret" otherwise he distributes "secret".

9. Player P_i counts how many votes of "secret" he got and sets $GSV\text{--}confidence_i$ of the Graded Share/Verify in the following manner:

   - *count of "secret"* $\geq 2t+1$ set $GSV\text{--}confidence_i = 2$, else,
   - *count of "secret"* $\geq t+1$ set $GSV\text{--}confidence_i = 1$, else,
   - otherwise $GSV\text{--}confidence_i = 0$.

Lemma 1.1 *If all honest players' polynomials do not define the same bivariate polynomial then each honest player P_i will set* GSV–confidence$_i = 0$.

Proof of Lemma

If at the end of step 5, $t+1$ or more honest players distributed "disqualify dealer" then all honest players will set $GSV\text{--}confidence_i = 0$. Thus we can assume that the number of satisfied honest players, at the end of Step 5, is $\geq t+1$. If P_i is satisfied that means that for all j, $g_i(j) = h_j(i)$. Let us assume, w.l.o.g., that $P_1, .., P_r$ $r \geq t+1$ are satisfied, and that they hold $\{g_1(y) \;\; h_1(x)\}, ..., \{g_r(y) \;\; h_r(x)\}$ respectively. Through the polynomials $g_1(y), ..., g_{t+1}(y)$ a single bivariate polynomial $\bar{f}(x,y)$, can be interpolated. From $\bar{f}(x,y)$ we can defin e $\bar{g}_i(y)$ and $\bar{h}_i(x)$ $\;1 \leq i \leq r$. We need to show that $\bar{g}_i(y) = g_i(y)$ and $\bar{h}_i(x) = h_i(x)$ for $1 \leq i \leq r$. And from this we can deduce that any subset of $t+1$ polynomials from this set define the same bivariate polynomial $\bar{f}(x,y)$. We immediately have that $\bar{g}_i(y) = g_i(y)$ for $1 \leq i \leq t+1$, from the definition.

Claim: For $1 \leq i \leq r$, $h_i(x) = \bar{h}_i(x)$.
It is enough to prove for $h_i(x)$ that $h_i(j) = \bar{h}_i(j)$ for $1 \leq j \leq t+1$, (because $h_i(x)$ is a polynomial of degree t, and if it is equal to another polynomial at $t+1$ points, then they are the same polynomial).
Proof of Claim
For $1 \leq j \leq t+1$

$$\bar{h}_i(j) \overset{\text{def of } \bar{h}}{=} \bar{f}(j,i) \overset{\text{def of } \bar{g}}{=} \bar{g}_j(i) \overset{\text{shown before}}{=} g_j(i) \overset{\text{def}}{=} h_i(j)$$

□

From the above we have that $h_1(x), ..., h_r(x)$ define $\bar{f}(x,y)$, and by the same reasoning $g_1(y), ..., g_r(y)$ define $\bar{f}(x,y)$. In other words all satisfied players define the same bivariate polynomial $\bar{f}(x,y)$. If at Step 6 the dealer gradecasts some polynomial $g_k(y)$ and $h_k(x)$ which do not satisfy the equation that $g_k(y) = \bar{g}_k(y)$ (same for h) then this polynomial will match at most t of the previously satisfied players thus increasing the number of unsatisfied players to $\geq t+1$. So either all $2t+1$ polynomials held by honest players define the same $\bar{f}(x,y)$ or they will all set their confidence to 0.

Lemma 1.2 *If the dealer is honest then for all honest players, P_i, we will have* GSV–confidence$_i$ = 2.

Proof of Lemma
This is equivalent to showing that no honest player will ever distribute "disqualify dealer". This can happen only if there is a contradiction between two values handed out by the dealer, which can never happen when the dealer is honest.

Recover Protocol

1. Player P_i distributes the polynomials $g_i(y)$ and $h_i(x)$.

2. Player P_i received w.l.o.g $g_1(y), h_1(x), ..., g_r(y), h_r(x)$ $\quad r \geq 2t+1$. He checks if $g_j(y)$ satisfies the equation

$$g_j(k_l) = h_{k_l}(j) \quad for \ \ l \geq 2t+1$$

If yes then he determines that $g_j(y)$ is in fact $f(j,y)$. He takes a set of $t+1$ good g's and interpolates through them to compute $f(x,y)$, and from that to compute $f(0,0)$.

4 Verifiable Secret Sharing

Theorem 2 *VSS can be achieved in constant expected time where $t < n/3$, with no broadcast channels and with no probability of error.*

We will start by stating our protocol.

VSS Protocol

First Phase:

1. Dealer executes Graded Share/Verify
2. All P_i's execute the expected constant-round [FeMi90] Byzantine Agreement where their input into the BA is as follows for P_i: if $GSV\text{−}confidence_i = 2$ then enter "yes" if $GSV\text{−}confidence_i = 0$ or 1then enter "no".
3. If result of BA is "yes" determine that there is a secret and that it is recoverable, otherwise the dealer is faulty.

Second Phase:

The Recover Protocol stated above.

Proof of Theorem:

The above protocol achieves Theorem 2.

Honest dealer: At the end of the GSV all honest players have $GSV\text{−}confidence_i = 2$, due to the property of "acceptance of good values", so they all enter "yes" into the BA. Because of the meaningfulness property of the BA, which states that if each honest player enters the same value, v, into the BA, then the result of the BA will be v, they will agree on "yes", achieving the desired properties of the VSS.

Dishonest dealer:

If at the end of the GSV the honest players have a $GSV\text{−}confidence_i =$ 0 or 1 then they will all enter "no" into the BA and as in the above case due to the meaningfulness of the BA the result of the BA will be "no". If the honest players have a $GSV\text{−}confidence_i = 1$ or 2 then some will enter "yes" and some "no". But in this case we don't mind what the result of the BA will be. If all honest players have $GSV\text{−}confidence_i = 1$ or 2, then due to the ***verifiability*** property of the GSV protocol they can all reconstruct the same secret. So whether they *all* decide to reconstruct or not they will be able to achieve their goal.

Run time: 15 steps
+ constant expect BA (FM)
O(1)

5 Concurrent VSS

Concurrent VSS Protocol - Is a distributed two phase protocol, executed by n players. Each player P_i holds a private value s_i. In the first phase of the protocol all players, concurrently, distribute their values. At the end of the first phase all honest players will determine for each player P_i whether he is disqualified, and if he is not then they all know that his value s_i can be recoverable in the second phase.

Theorem 3 *Concurrent VSS can be achieved in constant expected time where $t < n/3$, without the use of broadcast channels and with no probability of error.*

Concurrent VSS Protocol

1. Dealer $D_1, ..., D_n$ execute Graded Share/Verify concurrently, for values $s_1, ..., s_n$. Let us denote by $GSV\!-\!confidence_{ij}$ the confidence P_i has for the GSV executed by D_j.
2. Execute the expected constant-round Interactive Consistency Protocol of [BeEl88], where the value entered by P_i into the jth BA is: if $GSV\!-\!confidence_{ij} = 2$ then enter "yes", otherwise "no".
3. For all j, if $vector_i[j] = yes$ then player P_i determines that D_j's secret is recoverable, otherwise he determines that D_j is faulty.

Correctness: As for single VSS.
Runtime: 15 steps n independent VSS
+ constant expected n parallel BS (BE)
O(1)

6 Common Coin

Definition - A common coin is a coin which is visable to all players.

Main Theorem *A common coin for which $Pr(coin = 1) = 1/2$ can be achieved in constant expected time with no broadcast channels and $t < n/3$.*

Common Coin Protocol

1. All players P_i shares a random bit r_i using the Concurrent VSS Protocol
2. All players reconstruct the secrets which were not disqualified during the VSS. The set of secrets is $r_{i_1}, ..., r_{i_k}$ $k \geq 2t+1$
3. The coin will be $r_{i_1} \oplus ... \oplus r_{i_k}$

Claim The above protocol achieves our Main Theorem.
Proof The fact that the coin is common to all honest players is easily seen. Due to the BA they all consider the same subset of secrets as correct secrets, and so in step 2 they will all reconstruct the same set of secrets. Each reconstructed secret will be the same for all players because of the VSS properties. To see that the $Pr(coin = 1) = 1/2$ we need only note that there is at least one truly random bit shared by an honest player and that this bit is unknown to the dishonest players at the time when they commit to their value by sharing it using the VSS.

References

[BeEl88] M. Ben-Or and R. El-Yaniv. Interactive consistency in constant expected time. Inst. of Math. and Comp. Sci., Hebrew University, Jerusalem, 1988.

[BeGoWi88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for fault-tolerant distributed computing. In *Proc. 20th ACM Symposium on Theory of Computing*, pages 1–10, Chicago, 1988. ACM.

[BeMiRo90] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proc. 22th ACM Symposium on Theory of Computing*, May 1990.

[Bla79] G. Blakely. Safeguarding cryptographic keys. In *AFIPS*, volume 48, pages 313–317. NCC, June 1979.

[B.Mi86] S. Goldwasser B. Awerbuch, B. Chor and S. Micali. Verifiable secret sharing in the presence of faults. In *Proc. of the 27th Annual IEEE Symposium on Foundations of Computer Science*, 1986.

[BrDo84] A.Z. Broder and D. Dolev. Flipping coins in many pockets (byzantine agreement on uniformly random values. In *Proc. of the 25th Annual IEEE Symposium on Foundations of Computer Science*, pages 157–170. IEEE Computer Society Press, October 1984.

[ChCrDa88] D. Chaum, C. Crepeau, and I. Damgård. Multi-party unconditionally secure protocols. In *Proc. 20th ACM Symposium on Theory of Computing*, Chicago, 1988. ACM.

[FeMi90] P. Feldman and S. Micali. An optimal algorithm for synchronous byzantine agreement. Technical Report LCS/TM-425, MIT, June 1990. (Submitted for publication in SIAM J. on Computing.).

[GaHaYu87] Z. Galil, S. Haber, and M. Yung. Cryptographic computation: Secure falt-tolerant protocols and public-key model. In *Proc. CRYPTO 87*, pages 135–155. Springer Verlag, 1987.

[GoMiWi86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proc. of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 174–187, Toronto, 1986. IEEE.

[RaBe89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocoles with honest majority. In *Proc. 21th ACM Symposium on Theory of Computing*, 1989.

[Sha79] A. Shamir. How to share a secret. *Communcations of the ACM*, 22:612–613, November 1979.