Efficient Multiparty Computations Secure Against an Adaptive Adversary

Ronald Cramer[1], Ivan Damgård[2], Stefan Dziembowski[2], Martin Hirt[1], and Tal Rabin[3]

[1] ETH Zurich.[⋆]
{cramer,hirt}@inf.ethz.ch
[2] Aarhus University, BRICS[⋆⋆].
{ivan,stefand}@daimi.aau.dk
[3] IBM T.J.Watson Research Center
talr@watson.ibm.com

Abstract. We consider verifiable secret sharing (VSS) and multiparty computation (MPC) in the secure-channels model, where a broadcast channel is given and a non-zero error probability is allowed. In this model Rabin and Ben-Or proposed VSS and MPC protocols secure against an adversary that can corrupt any minority of the players. In this paper, we first observe that a subprotocol of theirs, known as weak secret sharing (WSS), is not secure against an adaptive adversary, contrary to what was believed earlier. We then propose new and adaptively secure protocols for WSS, VSS and MPC that are substantially more efficient than the original ones. Our protocols generalize easily to provide security against general Q^2-adversaries.

1 Introduction

Since the introduction of multiparty computation [Yao82,GMW87], its design and analysis has attracted many researchers, and has generated a large body of results. The problem stated very roughly is the following: Consider a set of players each holding a private input, who wish to compute some agreed upon function of their inputs in a manner which would preserve the secrecy of their inputs. They need to carry out the computation even if some of the players may become corrupted and actively try to interfere with the computation. Solutions to this problem have been given in various models and under different computational assumptions.

One of the major components of the model is the type of adversary which is assumed. The adversary is the entity which corrupts a set (of size up to t) of players during the execution of the protocol and takes control of their actions. Two types of adversaries have been considered in the literature (barring

[⋆] Supported by the Swiss National Science Foundation (SNF), SPP 5003-045293.
[⋆⋆] Basic Research in Computer Science, center of the Danish National Research Foundation.

slight variations): *static adversaries* and *adaptive adversaries*. The static adversary needs to choose the set of corrupted players before the execution of the protocol. The adaptive adversary on the other hand can choose the players during the execution of the protocol. It has been stated that the protocols of [BGW88,CCD88,RB89,Bea91] are secure against an adaptive adversary under the assumption that the players communicate via secure private channels.[1] In all these results the protocols are information theoretically secure. This has led many to believe that if a protocol is designed which is information theoretically secure and is executed in a model with private channels then the resulting protocol is immediately secure against an adaptive adversary. In the attempt to further our understanding of the power of these different adversaries we present an example of a natural protocol (which appears in [RB89]) which is information theoretically secure against a static adversary but fails against an adaptive adversary.

Another important goal in the design of these protocols is to provide protocols which are simple, so that they could actually be implemented in practice. For the case where the adversary can corrupt at most a third of the players reasonable protocols have been proposed [BGW88], but for the case where the adversary can corrupt a half of the players the existing solutions were quite cumbersome [RB89,Bea91]. In this paper we present solutions for multiparty computation (and for verifiable secret sharing) which are much more efficient than any existing protocol for the case where the adversary can corrupt up to a minority of the players.

More specifically we obtain a protocol for VSS which for probability of error $2^{-k+O(\log n)}$ with n players, requires $O((k+\log n)n^3)$ bits of communication as opposed to $\Omega((k+\log n)k^2n^4)$ bits required by existing protocols. This improvement is based in part on a more efficient implementation of *information checking protocol*, a concept introduced in [RB89] which can be described very loosely speaking as a kind of unconditionally secure signature scheme. Our implementation is linear meaning that for two values that can be verified by the scheme, any linear combination of them can also be verified with no additional information. This means that linear computations can be done non-interactively when using our VSS in MPC, contrary to the implementation of [RB89] (this property was also obtained in [Bea91], but with a less efficient information checking implementation).

An essential tool in MPC (provided in both [RB89] and [Bea91]) is a protocol that allows a player who has committed, in some manner, to values a, b, and c to show that $ab = c$ without revealing extra information. We provide a protocol for this purpose giving error probability 2^{-k} which is extremely simple. It allows a multiplication step in the MPC protocol to be carried out at cost equivalent to $O(n)$ VSS's, where all earlier protocols required $O(kn)$ VSS's.

Using methods recently developed in [CDM99], our protocols generalize easily to provide security against general Q^2-adversaries [HM97].

[1] The transformation of such protocols to the public channel model is outside the scope of this paper, but the interested reader can refer to [BH92,CFGN96].

Outline

We first show that the weak secret sharing (WSS) scheme of [RB89,Rab94] is not adaptively secure (Section 3). In Section 4, we propose an efficient implementation of information checking, and in Section 5, a scheme for verifiable secret sharing (VSS) is developed. Based on these protocols, in Section 6 an efficient protocol for multiparty computation (MPC) is presented. Finally, in Section 7 an efficient protocol secure against general (non-threshold) adversaries is sketched.

2 Model and Definitions

In this paper, we consider the *secure-channels model with broadcast*, i.e. there are n players $P_1, \ldots, P_n$ who are pairwise connected with perfectly private and authenticated channels, and there is a broadcast channel. There is a *central adversary* with unbounded computing power who actively corrupts up to t players where $t < n/2$. To actively corrupt a player means to take full control over that player, i.e. to make the player (mis)behave in an arbitrary manner. The adversary is assumed to be *adaptive* (or dynamic), this means that he is allowed to corrupt players during the protocol execution (and his choice may depend on data seen so far), in contrast to a static adversary who only corrupts players before the protocol starts. The security of the presented protocols is unconditional with some *negligible error probability*, which is expressed in terms of a *security parameter* k. The protocols operate in a finite field $K = GF(q)$, where $q > \max(n, 2^k)$.

2.1 Definition of Information Checking

Information checking (IC) is an information theoretically secure method for authenticating data. An IC scheme consists of three protocols:

$\mathsf{Distr}(D, INT, R, s)$ is initiated by the dealer D. In this phase D hands the secret s to the intermediary INT and some auxiliary data to both INT and the recipient R.

$\mathsf{AuthVal}(INT, R, s)$ is initiated by INT and carried out by INT and R. In this phase INT ensures that in the protocol RevealVal R (if honest) will accept s, the secret held by INT.

$\mathsf{RevealVal}(INT, R, s')$ is initiated by INT and carried out by INT and R. In this phase R receives a value s' from INT, along with some auxiliary data, and either accepts s' or rejects it.

The IC scheme has the following properties:

Correctness:

A. If D, INT, and R are uncorrupted, and D has a secret s then R will accept s in phase RevealVal.

B. If INT and R are honest then after the phases Distr and AuthVal INT knows a value s such that R will accept s in the phase RevealVal (except with probability 2^{-k}).

C. If D and R are uncorrupted, then in phase RevealVal with probability at least $1 - 2^{-k}$, player R will reject every value s' different from s.

Secrecy:

D. The information that D hands R in phase Distr is distributed independently of the secret s. (Consequently, if D and INT are uncorrupted, and INT has not executed the protocol RevealVal, R has no information about the secret s.)

Definition 1. *An IC scheme is a triple* (Distr, AuthVal, RevealVal) *of protocols that satisfy the above properties A. to D.*

2.2 Definition of WSS

An intuitive explanation for a *weak secret-sharing (WSS) scheme* is that it is a distributed analog of a computational commitment. A WSS scheme for sharing a secret $s \in K$ consists of the two protocols Sh and Rec. WSS exhibits the same properties, i.e. it binds the committer to a single value after the sharing phase Sh (this is equivalent to the commitment stage in the computational setting), yet the committer can choose not to expose this value in the reconstruction phase Rec (which is equivalent to the exposure of the commitments). WSS satisfies the following properties, with an allowed error probability 2^{-k}:

- *Termination*: If the dealer D is honest then all honest players will complete Sh, and if the honest players invoke Rec, then each honest player will complete Rec.
- *Secrecy*: If the dealer is honest and no honest player has yet started Rec, then the adversary has no information about the shared secret s.
- *Correctness*: Once all currently uncorrupted players complete protocol Sh, there exists a *fixed* value, $r \in K \cup \{\text{NULL}\}$, such that the following requirements hold:
  1. If the dealer is uncorrupted throughout protocols Sh and Rec then r is the shared secret, i.e. $r = s$, and each uncorrupted player will outputs r at the end of protocol Rec.
  2. If the dealer is corrupted then each uncorrupted player outputs either r or NULL upon completing protocol Rec.

Definition 2. *A t-secure WSS scheme for sharing a secret $s \in K$ is a pair* (Sh, Rec) *of two protocols that satisfy the above properties even in the presence of an active adversary who corrupts up to t players.*

2.3 Definition of VSS

An important protocol, which is widely used for multiparty computation, is verifiable secret sharing (VSS) [CGMA85]. In essence a VSS scheme allows a *dealer* to share a secret among n players in such a way that the adversary that corrupts at most t of the players, obtains no information about the secret. Furthermore, the secret can be efficiently reconstructed, even if the corrupted players try to disrupt the protocol. A more formal definition is the following:

A pair $(\mathsf{Sh}, \mathsf{Rec})$ of protocols is a *verifiable secret-sharing (VSS) scheme* if it satisfies a stronger correctness property, with an allowed error probability 2^{-k}:

– *Correctness*: Once all currently uncorrupted players complete protocol Sh, there exists a *fixed* value, $r \in K$, such that the following requirements hold:
  1. If the dealer is uncorrupted throughout protocol Sh then r is the shared secret, i.e. $r = s$, and each uncorrupted player outputs r at the end protocol Rec.
  2. If the dealer is corrupted then each uncorrupted player outputs r upon completing protocol Rec.

Definition 3. *A t-secure VSS scheme for sharing a secret $s \in K$ is a pair* $(\mathsf{Sh}, \mathsf{Rec})$ *of two protocols that satisfy the termination and the secrecy property of WSS, and the above, stronger, correctness property, even in the presence of an active adversary who corrupts up to t players.*

2.4 Definition of MPC

The goal of multiparty computation (MPC) is to evaluate an agreed function $g : K^n \rightarrow K$, where each player provides one input and receives the output. The privacy of the inputs and the correctness of the output is guaranteed even if the adversary corrupts any t players. For a formal definition for security see [GL90,MR91,Bea91,Can98,MR98].

3 Adaptive Security of WSS in [RB89]

In this section we describe a protocol which is secure against a static adversary yet fail against an adaptive one. The example captures nicely the power of the adaptive adversary to delay decisions and due to that cause different values to be computed during the protocol. The protocol which we examine is the weak secret-sharing scheme (WSS) of Rabin and Ben-Or [RB89,Rab94]. The attack will only work when $t > n/3$. It is important to note that this attack applies only to the WSS protocol of [RB89] as a stand-alone protocol, and does not apply to their VSS scheme, although it uses the WSS as a subprotocol.

In order to explain the attack we present a simplified protocol of the [RB89] protocol which assumes digital signatures. It is in essence the same protocol but with many complicating (non relevant) details omitted.

WSS Share (Sh)

The dealer chooses a random polynomial $f(x)$ of degree t, such that $f(0) = s$ the secret to be shared, and sends the share $s_i = f(i)$ with his signature for s_i to each player P_i.

WSS Reconstruct (Rec)

1. Every player reveals his share s_i and the signature on s_i.
2. If *all* properly signed shares $s_{i1}, \ldots, s_{ik}$ for $k \geq t$ interpolate a single polynomial $f'(x)$ of degree at most t, then the secret is taken to be $f'(0)$, otherwise no secret is reconstructed.

The definition of WSS requires that at the end of Sh a single value $r \in K \cup \{\mathsf{NULL}\}$ is set so that only that value (or NULL) will be reconstructed in Rec.

Clearly, if the adversary is static then the value r is set to the value interpolated through the shares held by the uncorrupted players. This value is well defined. If there exists a polynomial $f'(x)$ of degree t then $r = f'(0)$ otherwise r is NULL. During reconstruction if r was NULL then the players will set the output to NULL as all the shares of the good players will be considered in the interpolation and possibly some additional shares from the corrupted players. If r was not NULL then either the additional shares provided by the faulty players satisfy the polynomial $f'(x)$ in which case r will be reconstructed. But the adversary can decide to foil the reconstruction by having the corrupted players supply shares which do not match $f'(x)$, but this will only cause the players to output NULL but not another value $r' \neq r$.

Yet, we will show that under an adaptive adversary this requirement does not hold in the above described protocol. The attack for $n = 2t + 1$ proceeds as follows: In the protocol Sh the adaptive adversary corrupts the dealer causing him to deviate from the protocol. The dealer chooses two polynomials $f_1(x)$ and $f_2(x)$ both of degree at most t, where $f_1(0) \neq f_2(0)$, and $f_1(i) = f_2(i)$ for $i = 1, 2, 3$. For $i = 1, \ldots, 3$, player P_i receives the value $f_1(i)$ $(=f_2(i))$ as his share, for $i = 4, \ldots, t + 2$, player P_i receives $f_1(i)$, and for $i = t + 3, \ldots, 2t + 1$, player P_i receives $f_2(i)$ as his share. All shares are given out with valid signatures.

In Rec the adversary can decide whether to corrupt $P_4, \ldots, P_{t+2}$ thus forcing the secret to be $f_2(0)$, or to corrupt $P_{t+3}, \ldots, P_{2t+1}$ and thus force the secret to be $f_1(0)$. Hence it is clear that at the end of Sh there is not a *single* value which can be reconstructed in Rec. The decision on which value to reconstruct can be deferred by the adversary until the reconstruction protocol Rec is started.

Therefore the basic problem with stand-alone WSS is that it is not ensured that all honest players are on the same polynomial immediately after distribution. But when using it inside the VSS of [RB89], this property is ensured as a side effect of the VSS distribute protocol, hence the VSS protocol works correctly.

4 The Information Checking Protocol

In this section we present protocols that satisfy Definition 1 for information checking (cf. Section 2.1). They provide the same functionality as the check vector protocol from [RB89,Rab94] and the time capsule protocol from [Bea91]. However, our implementation of information checking also possesses an additional linearity property which will be utilized later in the paper.

The basic idea for the construction will be that the secret and the verification information will all lie on a polynomial of degree 1 (a line), where the secret will be the value at the origin. The dealer D hands to the intermediary INT two points on this line, and hands to the recipient R one point at a constant, but secret evaluation point α. This α is known to both D and R, but is unknown to INT. We will say that R will accept the secret which INT gives him only if the point which R holds lies on the line defined by the two points he receives from INT.

A general remark before we begin describing our protocols: In the following we adopt (for ease of exposition) the convention that whenever a player expects to receive a message from another player in the next step, and no message arrives, he assumes that some fixed default value was received. Thus we do not have to treat separately the case where no message arrives.

Definition 4. *A vector $(x, y, z) \in K^3$ is* 1_α-consistent *if there exists a degree 1 polynomial w over K such that $w(0) = x$, $w(1) = y$, $w(\alpha) = z$.*

Protocol $\mathsf{Distr}(D, INT, R, s)$**:**

The dealer D chooses a random value $\alpha \in K \setminus \{0, 1\}$ and additional random values $y, z \in K$ such that (s, y, z) is 1_α-consistent, and in addition he chooses a random 1_α-consistent vector (s', y', z'). D sends s, s', y, y' to the intermediary INT and z, z' to the recipient R.

Protocol Distr (together with RevealVal below) ensures ensures all properties except Property B. Adding the next protocol ensures this as well, without affecting A, C and D.

Protocol $\mathsf{AuthVal}(INT, R, s)$**:**

1. INT chooses a random element $d \in K$ and broadcasts $d, s' + ds, y' + dy$. If D observes that these values are incorrect, he broadcasts s, y. This counts as claiming that INT is corrupt. In this case the protocol ends here, and the broadcasted values will be used in the following. R will adjust his value for z, such that (s, y, z) is 1_α-consistent.
2. R checks if $(s' + ds, y' + dy, z' + dz)$ is 1_α-consistent. He broadcasts accept or reject accordingly. If D observes that R has acted incorrectly, he broadcasts z, α. This counts as claiming that R is corrupt. In this case the protocol ends here, and the broadcasted values will be used in the following. INT will adjust his value for y, such that (s, y, z) is 1_α-consistent.

3. If R rejected (and D did not claim him faulty) in the previous step, D must broadcast s, y, and the broadcasted values will be used in the following. R will adjust his value for z, such that (s, y, z) is 1_α-consistent.

Protocol $\mathsf{RevealVal}(INT, R, s)$**:**

1. INT broadcasts (s, y).
2. R verifies that (s, y, z) is 1_α-consistent and broadcasts accept or reject accordingly.

Lemma 1. *The protocols* (Distr, $\mathsf{AuthVal}$, $\mathsf{RevealVal}$) *described above satisfy Definition 1 for information checking (Section 2.1).*

Proof. We show that each property is satisfied:

A. It is clear that if all parties are honest, R will accept, and D will never broadcast any values.

B. The property is trivial in the cases where D broadcasts s, y or z, α. So it is enough to show that if D sends an inconsistent (s, y, z) initially, then R rejects with high probability. However, if for $e \neq d$, both $(s'+ds, y'+dy, z'+dz)$ and $(s'+es, y'+ey, z'+ez)$ are 1_α-consistent, then their difference and hence also (s, y, z) is 1_α-consistent. By the random choice of d it follows that R will accept with probability at most $1/|K|$ whenever (s, y, z) is inconsistent.

C. This property will follow from the fact that INT does not know α. Actually, we will show it holds, even if D uses the same α in all invocations of the protocol. We will exploit this property later. First note that INT learns no information on α from the $\mathsf{Distr}, \mathsf{AuthVal}$ protocols: what he gets in Distr has distribution independent of α. In $\mathsf{AuthVal}$, if he sends correct values, he knows in advance they will be accepted; if he doesn't, he knows that D will complain. Note also that this holds even if we consider many invocations of the authentication protocol together. Thus, all INT knows about α a priori is that it can be any value different from $0, 1$, and all candidates are equally likely.
Consider now the position of INT just before the opening of the first s-value. If he sends the correct s, y, or changes one of the values, he knows in advance R's reaction and so learns nothing new. If he sends s', y' where $s' \neq s, y' \neq y$, then R will accept if (s', y', z) is 1_α-consistent. We know that (s, y, z) is 1_α-consistent by its definition, thus so is $(s - s', y - y', 0)$. This gives a non-trivial degree 1 equation from which α can be computed. In other words, INT must guess α to have R accept a false value. He can do this with probability at most $1/(|K| - 2)$. On the other hand, if R rejects, all INT knows is that the solution to the equation is not the right value, so it can be excluded.
It follows by induction that if at most ℓ values are opened, at least $|K| - \ell - 2$ candidates for α remain from the point of view of INT, and no false values have been accepted, except with probability at most $\ell/(|K| - \ell - 2)$. In the application to VSS, ℓ will be linear in n, so the error probability is at most $2^{-k+O(\log n)}$.

D. If D and INT remain honest and R is corrupt, we must show that R does not learn s ahead of time. Observe that in the authentication protocol, R learns $z, z', d, s' + ds, y' + dy$. Note that since D and INT are honest, R knows in advance that $(s' + ds, y' + dy, z' + dz)$ will be 1_α-consistent. He can therefore compute $y' + dy$ from $z, z', d, s' + ds$, and this value can be deleted from his view without loss of generality. However, it is clear that $z, z', d, s' + ds$ has distribution independent of s.

Linearity of the IC Protocol
In our multiparty computation protocol we would like to be able to authenticate a linear combination of two values. The setting is as follows: D, R and INT have executed both protocols Distr and AuthVal for two different values s_1 and s_2. Now they wish to reveal a linear combination of these two secrets without exposing s_1 and s_2 and without carrying out any additional verification. This can be achieved if for both invocations of the IC protocol the dealer chooses *the same value* α as the random evaluation point which he gives to R. Then all the properties of the protocol still hold with the addition that the appropriate linear combination of the verification data yields a verification for the linear combination of s_1 and s_2.

IC-Signatures
In the sequel we will want to use the information checking protocol as semi "digital signatures". When a person receives a digital signature from a signer, he can later show it to anyone and have that person verify that it is in fact a valid signature. This property can be easily achieved with information checking, by carrying out the protocol with all players as explained bellow. We do not achieve all properties of digital signatures, but enough in order to achieve our goals.

The IC-Signatures will be given in the following way. Protocol Distr will be carried out by the dealer D with intermediary INT and the receiver being each player $P_1, \ldots, P_n$, each with respect to the same value s. Next, the AuthVal protocol will be performed by INT and each player P_i. Then, in protocol RevealVal, INT will broadcast s and the authentication information, and if $t + 1$ players accept the value s then we shall say that the "signature" has been confirmed. We shall call these signatures IC-signatures. These signature enable D to give INT a "signature" which only INT can use to convince the other players about the authenticity of a value received from the dealer. Thus, we use these IC-signatures as signatures given specifically from D to INT, and we denote such a signature as $\sigma_s(D, INT)$.

5 Verifiable Secret Sharing

We now present our simplified VSS protocol. The protocol is based on the bivariate solution of Feldman [FM88,BGW88] (omitting the need for error correcting codes). The protocol will use our new variant of information checking which will provide us with high efficiency.

Definition 5. *A vector* $(e_0, \ldots, e_{n-1}) \in K^n$ *is* t-consistent *if there exists a polynomial* $w(x)$ *of degree at most* t *such that* $w(i) = e_i$ *for* $0 \le i < n$.

The intuition behind the construction is that the secret will be shared using an $n \times n$ matrix of values, where each row and column is t-consistent, and where row and column i is given to player P_i. Thus, for $i \neq j$, P_i and P_j share two values in the matrix. The dealer will commit himself to all the values by signing each entry in the matrix. The row determines by simple interpolation a share of a single variate polynomial. Thus, de facto the dealer has given player P_i a signed share, s_i. The players can now check consistency of the matrix by comparing values between them and expose inconsistent behavior by the dealer using the signatures. Hence we are guaranteed that all the values held by (yet) uncorrupted players are consistent and define a single secret.[2] In order to also have the share of player P_i signed (implicitly) by the other players, player P_i gets the share b_{ij} in his row signed by player P_j. Now this in return will prevent the adversary from corrupting the secret at reconstruction time.

VSS Share (Sh)

1. The dealer D chooses a random bivariate polynomial $f(x,y)$ of degree at most t in each variable, such that $f(0,0) = s$. Let $s_{ij} = f(i,j)$. The dealer sends to player P_i the values $a_{1i}{=}s_{1i}, \ldots, a_{ni}{=}s_{ni}$ and $b_{i1}{=}s_{i1}, \ldots, b_{in}{=}s_{in}$. For each value a_{ji}, b_{ij} D attaches a digital signature $\sigma_{a_{ji}}(D, P_i)$, $\sigma_{b_{ij}}(D, P_i)$.
2. Player P_i checks that the two sets $a_{1i}, \ldots, a_{ni}$ and $b_{i1}, \ldots, b_{in}$ are t-consistent. If the values are not t-consistent, P_i broadcasts these values with D's signature on them. If a player hears a broadcast of inconsistent values with the dealer's signature then D is disqualified and execution is halted.
3. P_i sends a_{ji} and a signature which he generates on a_{ij}, $\sigma_{a_{ji}}(P_i, P_j)$ privately to P_j.
4. Player P_i compares the value a_{ij} which he received from P_j in the previous step to the values b_{ij} received from D. If there is an inconsistency, P_i broadcasts $b_{ij}, \sigma_{b_{ij}}(D, P_i)$.
5. Player P_i checks if P_j broadcasted a value $b_{ji}, \sigma_{b_{ji}}(D, P_j)$ which is different than the value a_{ji} which he holds. If such a broadcast exists then P_i broadcasts $a_{ji}, \sigma_{a_{ji}}(D, P_i)$.
6. If for an index pair (i,j) a player hears two broadcasts with signatures from the dealer on different values, then D is disqualified and execution is halted.

VSS Reconstruct (Rec)

1. Player P_i broadcasts the values $b_{i1}, \ldots, b_{in}$ with the signature for value b_{ij} which he received from player P_j. (If he did not receive a signature from P_j in the protocol Sh then he had already broadcasted that value with a signature from D.)

[2] So far, this results in a WSS which is secure against an adaptive adversary.

2. Player P_i checks whether player P_j's shares broadcasted in the previous step are t-consistent and all the signatures are valid. If not then P_j is disqualified.
3. The values of all non-disqualified player are taken and interpolated to compute the secret.

Theorem 1. *The above protocols* (Sh, Rec) *satisfy Definition 3 for VSS protocols.*

Proof. We prove that each required property is satisfied:

Secrecy. Observe that in Steps 2–6, the adversary learns nothing that he was not already told in Step 1. Thus the claim follows immediately from the properties of a bi-variate polynomial of degree t and the properties of the information checking.

Termination. From examining the protocol it is clear that the dealer D can be disqualified only if the data which he shared is inconsistent, assuming that the players cannot forge any of the dealers signatures, of which there are $O(n)$. Thus, an honest dealer will be disqualified at most with probability $O(2^{-k+\log n})$.

Correctness. First we will show that a fixed value r is defined by the distribution. Define r to be the secret which interpolates through the shares held by the set of the first $t+1$ players who have not been corrupted during Sh. Their shares are trivially t-consistent, and with probability at least $1-O(2^{-k+\log n})$, there are correct signatures for these shares, and thus they define uniquely an underlying polynomial $f'(x,y)$ as well as a secret $r = f'(0,0)$. Let us look at another uncorrupted player outside this set. He has corroborated his shares with all these $t+1$ players and has not found an inconsistency with them. Moreover, this player has also verified that his row and column are t-consistent. Hence, when this player's shares are added to the initial set of players' shares the set remains t-consistent, thus defining the same polynomial f' and secret r. Now we examine the two correctness conditions:
1. It is easy to see that if D is uncorrupted then this value $r = s$.
2. A value different than r will be interpolated (or the reconstruction will fail) only if a corrupted player would be able to introduce values which are inconsistent with the values held by the honest players. A corrupted player succeeded doing it only when he was not disqualified in Step 2. of the reconstruction procedure. This means that he was able to produce a set of n values which are t-consistent, and for each value to have a signature from the appropriate player to which it relates. Clearly, $t+1$ of these signatures must be from still uncorrupted players. We have already shown that these players' shares lie on $f'(x,y)$, thus if the corrupted player's shares are t-consistent they must lie on $f'(x,y)$ as well. Therefore the adversary cannot influence the value of the revealed secret.

□

Efficiency. By inspection of the VSS distribution protocol Sh, one finds that n^2 field elements are distributed from D, and each of these are authenticated using Distr and AuthVal a constant number of times. Executing Distr and AuthVal requires communicating a constant number of field elements for each player, and so we find that the total communication is $O((k + \log n)n^3)$ bits, for an error probability of $2^{-k+O(\log n)}$.

6 Multiparty Computation

Based on the VSS scheme of the previous section, we now build a multiparty computation protocol. Based on the [BGW88] paradigm it is known that it is sufficient to devise methods for adding and multiplying two shared numbers.

Note that in our case (contrary to e.g. [BGW88]) a VSS of a value a consists not only of the shares $a_1, \ldots, a_n$ where a_i is held (in fact implicitly) by P_i, it is explicitly held by P_i via the subshares $a_{i1}, \ldots, a_{in}$ where a_{ij} is held also by player P_j, and P_i has a IC-signature from P_j on that value. This structure and the IC-signatures are required for the reconstruction. Thus, if we wish to compute the sum/multiplication of two secrets we need to have the resultant in this same form.

We will prove the following theorem in the next two subsections.

Theorem 2. *Assume the model with a complete network of private channels between n players and a broadcast channel. Let C be any arithmetic circuit over the field K, where $|K| > max(n, \log k)$ and k is a security parameter. Then there is a multiparty computation protocol for computing C, secure against any adaptive adversary corrupting less than $n/2$ of the players. The complexity of this protocol is $O(n^2|C|)$ VSS protocols with error probability $2^{-k+O(\log n)}$, where $|C|$ is the number of gates in C. This amounts to $O(|C|kn^5)$ bits of communication.*

6.1 Addition

Addition is straightforward: For two secrets a and b shared with (implicit) shares $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, all the subshares, and their appropriate IC-signatures, each player P_i needs to add his two (implicit) shares a_i and b_i which means that he needs to hold a IC-signature from P_j for $a_{ij} + b_{ij}$. But this is immediately achieved as the sum of two IC-signatures results in an IC-signature for the sum of the values signed. Thus, we have computed the addition of two shared secrets.

6.2 Multiplication

Multiplication is slightly more involved. Assume that we have two secrets a and b with (implicit) shares $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ and all the subshares and their appropriate IC-signatures. We apply the method from [GRR98]. This method calls for every player to multiply his shares of a, resp. b and to share the result of this using VSS. This results in n VSS's and a proper sharing of the result c can be computed as a fixed linear combination of these (i.e. each player computes a linear combination of his shares from the n VSS's). Since our VSS is linear, like the one used in [GRR98], the same method will work for us, provided we can show that player P_i can share a secret c_i using VSS, such that it will hold that $c_i = a_i b_i$ and to prove that he has done so properly. If P_i fails to complete this process the simplest solution for recovery is to go back to the start of the computation, reconstruct the inputs of P_i, and redo the computation, this time simulating P_i openly. This will allow the adversary to slow down the computation by at most a factor linear in n.

In order to eliminate subindices let us recap our goal stated from the point of view of a player D. He needs to share a secret c using VSS which satisfies that $c = ab$. The value a is shared via subshares $a_1, \ldots, a_n$ (lying on a polynomial f_a, say) where a_i is held by player P_i and D holds an IC-signature of this value from P_i. The same holds for the value b (with a polynomial f_b).

1. D shares the value $c = ab$ using the VSS Share protocol. Let f_c be the polynomial defined by this sharing.[3]
2. D chooses a random $\beta \in K$ and he shares β and βb. The sharing of these values is very primitive. D chooses a polynomial $f_\beta(x) = \beta_t x^t + \ldots + \beta_1 x + \beta$ and gives player P_i the value $f_\beta(i)$ and an IC-signature on this value. A player complains if he did not receive a share and a signature, and the dealer exposes these values. The same is done for βb (with a polynomial $f_{\beta b}$).
3. The players jointly generate, using standard techniques, a random value r, and expose it.
4. D broadcast the polynomial $f_1(x) = r f_a(x) + f_\beta(x)$.
5. Player P_i checks that the appropriate linear combination of his shares lies on this polynomial, if it does not he exposes his signed share $f_\beta(i)$ and requires the dealer to expose the IC-signature which the dealer holds generated by P_i for the value a_i. If the dealer fails then D is disqualified.
6. If the dealer has not been disqualified each player locally computes $r_1 = f_1(0)$.
7. D broadcasts the polynomial $f_2(x) = r_1 f_b(x) - f_{\beta b}(x) - r f_c(x)$.
8. Each player checks that the appropriate linear combination of his shares lies on this polynomial, if it does not he exposes his signed share $f_{\beta b}(i)$ and $f_c(i)$ and requires the dealer to expose the IC-signature which the dealer holds generated by P_i for the value b_i. If the dealer fails then D is disqualified.
9. If D has not been disqualified P_i verifies that $f_2(0) = 0$, and accepts the sharing of c, otherwise D is disqualified.

The security of the protocol is guaranteed by the following lemma.

Lemma 2. *Executing the above protocol for sharing $c = ab$ does not give the adversary any information that he did not know before.*

Proof. Wlog we can assume that the dealer is honest. Thus all the values revealed during the protocol look random to the adversary (except for the polynomial f_2 which is a random polynomial such that $f_2(0) = 0$). Therefore the adversary learns nothing. $\square$

[3] Note that f_c is not the bivariate polynomial directly constructed by D rather it is the univariate polynomial defined by the implicit shares of c.

Lemma 3. *If $c \neq ab$ in the above protocol, then the probability that the dealer succeeds to perform the above is at most $\frac{1}{|K|}$.*

Proof. Suppose there exist two distinct challenges r_1 and r_1' such that if any of them is chosen in Step 3. then D is not disqualified in the next rounds. Step 4. guarantees that honest players have consistent shares of f_β, since we open f_1 and we know f_a is consistent. So there is a well-defined value β shared by f_β. In the same way Step 7 guarantees that $f_{\beta b}$ is consistent, so it defines some value z (which may or may not be βb). Now from Step 4., $r_1 = ra + \beta$ and $r_1' = r'a + \beta$, so from Step 7., we get $(ra+\beta)b + z + rc = 0 = (r'a+\beta)b + z + r'c$ and we conclude that $ab = c$. □

7 General Adversaries

It is possible to go beyond adaptive security against any dishonest *minority*, by considering *general*, i.e. not necessarily threshold *adversaries* [HM97]. The corruption capability of such an adversary is specified by a family of subsets of the players, where the adversary is restricted to corrupting one of these sets - dishonest minority is clearly a special case. Our results in this paper extend to the general scenario, following ideas developed in [CDM99].

First, by replacing Shamir secret sharing by monotone span program (MSP) secret sharing [KW93] in our VSS, we immediately obtain WSS protocols secure against any Q^2-adversary [HM97], with communication and computation polynomial in the monotone span program complexity of the adversary [CDM99]. A Q^2-adversary is an adversary who is capable of corrupting only subsets of players in a given family of subsets, where no two subsets in the family together cover the full player set.

The reason why the generalized protocol is only a WSS and not a VSS is that for a general linear secret sharing scheme, a qualified subset of shares define uniquely the secret, but NOT necessarily the entire set of shares (in contrast with what is the case for Shamir's threshold scheme).

However, building on the linearity of this WSS and monotone span program secret sharing, we can still construct efficient VSS (with negligible, but non-zero error) secure against any Q^2-adversary.

Roughly speaking, the idea (taken from [CDM99]) is that the dealer will use WSS to commit to his secret and the set of shares. He can then convince the players that this was done correctly. This amounts to showing a number of linear relations on committed values, which is easy by linearity of the WSS. Finally, each commitment to a share is privately opened to the player that is to receive it.

The resulting VSS enables multi-party computation secure against any Q^2-adversary if we base the construction of VSS on a so called MSP with multiplication [CDM99]. Such an MSP always exists, and can be chosen to have size at most twice that of a minimal MSP secure against the adversary. As far as general adversaries are concerned, security against Q^2-adversaries is the maximum attainable level of security.

This construction gives a VSS with complexity $O((k+\log n)nm^3)$ bits, where m is the size of the monotone span program. In some independent work Smith and Stiglic[SS98] present a somewhat similar idea, which however results in a less efficient protocol ($O(k^2(k+\log n)nm^3)$ bits) because they directly apply the ideas from [CDM99] to [Rab94], i.e. replace in [Rab94] Shamir's secret sharing by the monotone span programs with multiplication from [CDM99].

Acknowledgment. We are very grateful to Adam Smith and Anton Stiglic for pointing out an error in the information checking protocols of the almost final version of this paper.

References

[Bea91] D. Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4:75–122, 1991.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In 20*th STOC*, pp. 1–10. ACM, 1988.

[BH92] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. *Eurocrypt '92*, pp. 307–323. Springer LNCS 658, 1992.

[Can98] R. Canetti. Security and composition of multiparty cryptographic protocols. Manuscript, to appear, 1998.

[CCD88] D. Chaum, C. Crepeau, and I. Damgård. Multiparty unconditionally secure protocols. In 20*th STOC*, pp. 11–19. ACM, 1988.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *26th FOCS*, pp. 383–395. IEEE, 1985.

[CDM99] R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret-sharing scheme. Manuscript, 1999.

[CFGN96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In 28*th STOC*, pp. 639–648. ACM, 1996.

[FM88] P. Feldman and S. Micali. An optimal algorithm for synchronous Byzantine agreement. In 20*th STOC*, pp. 148–161. ACM, 1988.

[GL90] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. *Crypto '90*, pp. 77–93. Springer LNCS 537, 1990.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In 19*th STOC*, pp. 218–229. ACM, 1987.

[GRR98] R. Gennaro, M. Rabin, and T Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In 17*th PODC*, pp. 101–111. ACM, 1998.

[HM97] M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in general multiparty computations. In 16*th PODC*, pp. 25–34. ACM, 1998.

[KW93] M. Karchmer and A. Wigderson. On span programs. In *Proc. of Structure in Complexity*, pp. 383–395, 1993.

[MR91] S. Micali and P. Rogaway. Secure computation. *Crypto '91*, pp. 392–404. Springer LNCS 576, 1991.

[MR98] S. Micali and P. Rogaway. Secure computation: The information theoretic case. Manuscript, to appear, 1998.

[Rab94] T. Rabin. Robust sharing of secrets when the dealer is honest or faulty. *Journal of the ACM*, 41(6):1089–1109, 1994.

[RB89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In 21*st STOC*, pp. 73–85. ACM, 1989.

[SS98] A. Smith and A. Stiglic. Multiparty computations unconditionally secure against Q^2 adversary structures. Manuscript, 1998.

[Yao82] A.C. Yao. Protocols for secure computations. In 23*rd FOCS*, pp. 160–164. IEEE, 1982.