An Error in the Mixed Adversary Protocol by Fitzi, Hirt and Maurer

Ivan Damgård*, Aarhus University, BRICS†

February 10, 1999

Abstract

We point out an error in the protocol for mixed adversaries and zero error from the Crypto 98 paper by Fitzi, Hirt and Maurer. We show that the protocol only works under a stronger requirement on the adversary than the one claimed. Hence the bound on the adversary's corruption capability given there is not tight. Subsequent work has shown, however, a new bound which is indeed tight.

1 Introduction

In [1], Fitzi, Hirt and Maurer present a treatment of mixed adversaries against multiparty computation protocols (in the model with private channels and unconditional security). Such an adversary can actively corrupt t_a of the n players, meaning that he takes full control over them, and can additionally passively corrupt t_p players, meaning that he gets access to all their data, but they continue to follow the protocol. They also consider adversaries that can further fail corrupt some players, simply making them stop playing, but we do not need to consider this for the purpose of this note.

The goal of [1] is to give bounds on t_p, t_a necessary and sufficient to enable general multiparty computation. In particular, for the case of perfect

*This research was carried out while the author was visiting the IBM T.J. Watson research laboratories

†Basic Research In Computewr Science, Center of the Danish National Research Foundation

protocols (i.e. with zero error probability), it is claimed that the condition $3t_a + t_p < n$ and $2t_a + 2t_p < n$ is necessary and sufficient.

In this short note, we point out that this result is not correct. The condition is not sufficient: the protocol given in [1] to show it suffices in fact only works when the stronger $3t_a + 2t_p < n$ is satisfied.

This is no coincidence: having been informed about the error, Fitzi, Hirt and Maurer have managed to show that in fact $3t_a + 2t_p < n$ is necessary; or more precisely: there are functions that cannot be computed securely against an adversary violating the condition.

Note that nothing in this note concerns the results in [1] about protocols with non-zero error. To the best of our knowledge, those results are correct.

2 The Protocol and the Problem

The protocol given in [1] is a natural generalization of the BGW protocol [2]: since the number of players from which the adversary can get information is $t_a + t_p$, we will share information using polynomials of degree $d = t_a + t_p$. The verifiable secret sharing protocol of BGW can then be excuted as is, except that the rule for disqualifying a dealer in this case will be that he is out if more than t_a players accuse him. It is argued (correctly) in [1] that this works if (and only if) $3t_a + t_p < n$.

The problem arises in the protocol for multiplying two shared values. This is based on the usual idea of having the players locally multiply values of the two polynomials used to share the values. Clearly, we need to have that $2d = 2t_a + 2t_p < n$ for this to work at all: otherwise there would not be enough players to determine the resulting polynomial (of degree $2d$).

In order to ensure that the multiplication works correctly when $t_a > 0$, players need to prove that they perform their local multiplications correctly. And it is in the subprotocol for this purpose that the problem occurs. Assume that a player D has shared values a, b using polynomials $f(x), g(x)$, so that $a = f(0), b = g(0)$ and player i holds $f(i), g(i)$. We need to generate a new polynomial (of degree d) that is guaranteed to have value $c = ab$ in 0.

If we define $h(x) = f(x)g(x)$, then $h(0) = ab$, and each player can compute $h(i) = f(i)g(i)$. Of course, h has degree $2d$. However, player D knows all of $h(x)$, and it is shown in [2] how to exploit this: D will choose a random polynomial $h_1(x)$ of degree d, and with the same leading coefficient as $h(x)$. He will distribute it using the VSS protocol, such that all players can verify that it has degree d. Then player i can compute $h(i) - i^d h_1(i)$ which is the value of $h(x) - x^d h_1(x)$ in point i. This polynomial has degree $2d-1$. The same method can be applied again, and so after subtracting shifted versions of $h_1(x), ..., h_d(x)$, we end up with a polynomial that is guaranteed to have the same value in 0 as h, and can be verified to have degree d.

Consider what happens to this protocol when applied in a situation where we assume only that $3t_a + t_p < n$ and $2t_a + 2t_p < n$. Note that in some cases, we can still have $3t_a + 2t_p \geq n$, and we will argue that in these cases, the above subprotocol fails. For concreteness, let us think of the case where $2t_p + 2t_a = n - 1$, and $t_a > 0$.

The adversary can make the protocol fail whenever D is actively corrupted, as follows: when players have multiplied their values of $f(x)$ and $g(x)$, the actively corrupted players will simply change their minds about their value of $g(i)h(i)$ and replace it by another value (which in this case can be chosen arbitrarily). Since here $2d = n - 1$, any set of n values is consistent with some polynomial of degree $2d$. Let $h'(x)$ be the new polynomial. Clearly, the adversary can choose h' such that $h'(0)$ is any value he desires. Now all that remains is for D to choose his polynomials $h_1(x), ..., h_d(x)$ such that they will correctly reduce the degree of the new polynomial $h'(x)$ to degree d. An honest player will not be able to tell the difference between this and an honest D.

Clearly this attack works whenever the set of players following the protocol is too small to determine uniquely a polynomial of degree $2d$, i.e. when $n - t_a \leq 2d = 2t_a + 2t_p$. In other words, the attack fails precisely when $3t_a + 2t_p < n$. It is not hard to show that in this case not only does the attack fail, the multiplication protocol in is fact secure.

Since this subprotocol is the only one that does not work also under the weaker condition, we see that the condition $3t_a + 2t_p < n$ is *sufficient* for general multiparty computation. In the Crypto version of [1], it is only proved that $3t_a + t_p < n$ and $2t_a + 2t_p < n$ is necessary. Fortunately, this is not an optimal result: having been informed about the mistake, the authors of [1] have managed to show that in fact $3t_a + 2t_p < n$ is also *necessary*. For completeness, we include here a sketch of their proof . More details will appear in an upcoming final version of [1].

In order to prove the necessity of $3t_a + 2t_p < n$, assume for contradiction that for some t_a, t_p with $3t_a + t_p \geq n$ every function can be computed perfectly (t_a, t_p)-securely. Then one can construct a protocol for three processors $\hat{p}_1$, $\hat{p}_2$, and $\hat{p}_3$, where $\hat{p}_1$ plays for $t_a + t_p$ processors, $\hat{p}_2$ plays for $t_a + t_p$ other processors, and $\hat{p}_3$ plays for the remaining at most t_p processors. This new protocol is secure with respect to an adversary that passively corrupts either $\hat{p}_1$ or $\hat{p}_2$, or actively corrupts $\hat{p}_3$.

It follows that there exists a protocol secure against such an adversary for computing the logical `AND` of two bits x_1 and x_2 held by $\hat{p}_1$ and $\hat{p}_2$, respectively. Let T denote the transcript of the broadcast channel of a run of that protocol (if no broadcast channel is available, let $T = \emptyset$), and let T_{ij} $(1 \leq i < j \leq 3)$ denote the transcript of the channels between $\hat{p}_i$ and $\hat{p}_j$. Due to the requirement of perfect privacy, $\hat{p}_1$ will not send any information about his bit x_1 over T_{12} or over T before he knows x_2 (if P_1 knows that $x_2 = 1$ he can reveal x_1). Similarly, $\hat{p}_2$ will not send any information about x_2 over T_{12} or over T before he knows x_1. Hence the only escape from this deadlock would be to use $\hat{p}_3$. However, as T_{12} and T jointly give no information about x_2, a random misbehavior of an actively corrupted $\hat{p}_3$ (ignore all received messages and send random bits whenever a message must be sent) would with some (possibly negligible) probability make $\hat{p}_1$ receive the wrong output, contradicting the perfect security of the protocol.

References

[1] M. Fitzi, M. Hirt and U. Maurer: *Trading Privacy for Correctness in Multiparty Computation*, Proceedings of Crypto 98, Springer Verlag LNCS series.

[2] M. Ben-Or, S. Goldwasser, A. Wigderson: *Completeness theorems for Non-Cryptographic Fault-Tolerant Distributed Computation*, Proc. ACM STOC '88, pp. 1–10.