A Note on the Relation between the Definitions of Security for Semi-Honest and Malicious Adversaries*

Carmit Hazay† Yehuda Lindell‡

October 28, 2010

Abstract

In secure computation, a set of parties wish to jointly compute some function of their private inputs while preserving security properties like privacy, correctness and more. The two main adversary models that have been considered are *semi-honest* adversaries who follow the prescribed protocol but try to glean more information than allowed from the protocol transcript, and *malicious* adversaries who can run any efficient strategy in order to carry out their attack. As such they can deviate at will from the prescribed protocol. One would naturally expect that any protocol that is secure in the presence of malicious adversaries will automatically be secure in the presence of semi-honest adversaries. However, due to a technicality in the definition, this is not necessarily true. In this brief note, we explain why this is the case, and show that a slight modification to the definition of semi-honest adversaries (specifically, allowing a semi-honest adversary to change its received input) suffices for fixing this anomaly.

Our aim in publishing this note is to make this curious fact more known to the wider cryptographic community.

1 Malicious Versus Semi-honest Adversaries

In order to keep this note brief, we assume that the reader is familiar with the *exact* definitions of relevance. We refer to [2], [4, Chapter 7], or [5, Chapter 2] for motivation and full definitions of secure computation in the presence of semi-honest and malicious adversaries.

At first sight, it seems that any protocol that is secure in the presence of malicious adversaries is also secure in the presence of semi-honest adversaries. This is because a semi-honest adversary is just a "special case" of a malicious adversary who faithfully follows the protocol specification. Although this is what we would expect, it turns out to be *false*. This anomaly is due to the fact that although a real semi-honest adversary is indeed a special case of a real malicious adversary, this is not true of the respective adversaries in the ideal model. Specifically, the adversary in the ideal model for malicious adversaries is allowed to change its input, whereas the adversary in the ideal model for semi-honest adversaries is not. Thus, the adversary/simulator for the case of malicious adversaries has more power than the adversary/simulator for the case of semi-honest adversaries. As such, it may be possible to simulate a protocol in the malicious model, but not in the semi-honest model. We now present two examples of protocols where this occurs.

*We thank Yuval Ishai for first pointing out this inconsistency in the definitions to us. Most of this note is an excerpt from [5].

†Dept. of Computer Science, Aarhus University, Denmark. carmit@cs.au.dk.

‡Dept. of Computer Science, Bar-Ilan University, Israel. lindell@cs.biu.ac.il.

Example 1 – secure AND. Consider the case of two parties computing the *binary AND* function $f(x, y) = x \wedge y$, where only party P_2 receives output. Note first that if party P_2 uses input 1, then by the output received it can fully determine party P_1's input (if the output is 0 then P_1 had input 0, and otherwise it had input 1). In contrast, if party P_2 uses input 0 then it learns nothing about P_1's input, because the output equals 0 irrespective of the value of P_1's input. The result of this observation is that in the ideal model, an adversary corrupting P_2 can always learn P_1's exact input by sending the trusted party the input value 1. Thus, P_1's input is always revealed. In contrast, in the ideal model with a semi-honest adversary, P_1's input is only revealed if the corrupted party has input 1; otherwise, the adversary learns nothing whatsoever about P_1's input. We use the above observations to construct a protocol that securely computes the binary AND function in the presence of malicious adversaries, but is not secure in the presence of semi-honest adversaries; see Protocol 1.

PROTOCOL 1 (A Protocol for Binary AND)

- **Input:** P_1 has an input bit x and P_2 has an input bit y.
- **Output:** The binary value $x \wedge y$ for P_2 only.
- **The protocol:**
  1. P_1 sends P_2 its input bit x.
  2. P_2 outputs the bit $x \wedge y$.

We have the following claims:

Claim 2 *Protocol 1 securely computes the binary AND function in the presence of malicious adversaries.*

Proof: We separately consider the case where P_1 is corrupted and the case where P_2 is corrupted. If P_1 is corrupted, then the simulator $\mathcal{S}$ (for malicious adversaries) receives from $\mathcal{A}$ the bit that it sends to P_2 in the protocol. This bit fully determines the input of P_1 to the function and so $\mathcal{S}$ just sends it to the trusted party, thereby completing the simulation. In the case where P_2 is corrupted, $\mathcal{S}$ sends input 1 to the trusted party and receives back an output bit b. By the observation above, b is the input of the honest P_1 in the ideal model. Thus, the simulator $\mathcal{S}$ just hands $\mathcal{A}$ the bit $x = b$ as the value that $\mathcal{A}$ expects to receive from the honest P_1 in a real execution. It is immediate that the simulation here is perfect. ■

We stress that the above works because P_2 is the only party to receive output. If P_1 also were to receive output, then $\mathcal{S}$'s simulation in the case of a corrupted P_2 would not work. In order to see this, consider an adversary who corrupts P_2, uses input $y = 0$ and outputs its view in the protocol, including the bit x that it receives from P_1. In this case, $\mathcal{S}$ cannot send $y = 1$ to the trusted party because P_1's output would not be correctly distributed. Thus, it must send $y = 0$, in which case the view that it generates for $\mathcal{A}$ cannot always be correct because it does not know the input bit x of P_1.

Claim 3 *Protocol 1 does not securely compute the binary AND function in the presence of semi-honest adversaries.*

Proof: Consider the simulator S_2 (for semi-honest adversaries) that is guaranteed to exist for the case where P_2 is corrupted. Then, S_2 is given y and $x \wedge y$ and must generate the view of P_2 in the computation. However, this view contains the value x that P_1 sends to P_2 in the protocol. Now, if $y = 0$ and x is random, then there is no way that S_2 can guess the value of x with probability greater than $1/2$. We conclude that the protocol is not secure in the presence of semi-honest adversaries. ■

We remark that in fact, the AND function for the case of semi-honest adversaries is *complete*, and as such, implies the existence of oblivious transfer [1]. In contrast, as we have shown, in the case of malicious adversaries it can be securely computed unconditionally, without any hardness assumptions. As such it is *trivial*.

This leads to the following remarkable corollary:

Corollary 4 *There exists a binary function (where only one party receives output) that is* trivial *in the presence of malicious adversaries and* complete *in the presence of semi-honest adversaries.*

Example 2 – set union. Another example where this arises is the problem of set union over a large domain where only one party receives output. Specifically, consider the function $f(X, Y) = (\lambda, X \cup Y)$ where $X, Y \subseteq \{0,1\}^n$ are sets of the same size, and λ denotes the "empty" output. We claim that the protocol where P_1 sends its set X to P_2 is secure in the presence of malicious adversaries. This follows for the exact same reasons as above because a corrupted P_2 in the malicious model can replace its input set Y with a set Y' of the same size, but containing random values. Since the sets contain values of length n, it follows that the probability that $X \cap Y \neq \phi$ is negligible. Thus, the output that P_2 receives completely reveals the input of P_1. In contrast, if a corrupted party cannot change its input, then when $X \cap Y \neq \phi$ the elements that are common to both sets are hidden. Specifically, if five elements are common to both sets, then P_2 knows that there are five common elements, but does not have any idea as to which are common. Thus, for the same reasons as above, the protocol is not secure in the presence of semi-honest adversaries. Once again, we stress that this works when only one party receives output; in the case where both parties receive output, securely computing this functionality is highly non-trivial.

Discussion. It is our opinion that the above phenomenon should not be viewed as an "annoying technicality". Rather it points to a problem in the definitions that needs to be considered. Our position is that it would be better to define semi-honest adversaries as adversaries that *are allowed to change their input* before the computation starts (e.g., by rewriting the value on their input tape), and once the computation begins must behave in a semi-honest fashion as before. Conceptually, this makes sense because parties are allowed to choose their own input and this is not adversarial behavior. In addition, this model better facilitates the "compilation" of protocols that are secure in the semi-honest model into protocols that are secure in the malicious model. Indeed, in order to prove the security of the protocol of [3], and specifically the compilation of a protocol for the semi-honest model into one that is secure in the presence of malicious adversaries, Goldreich introduces the notion of augmented semi-honest behavior, which is exactly as described above; see Definition 7.4.24 in Section 7.4.4.1 of [4]. We stress that all protocols presented in this book that are secure in the presence of semi-honest adversaries are also secure in the presence of *augmented semi-honest adversaries*. Furthermore, as stated in the following proposition, security in the malicious model implies security in the augmented semi-honest model, as one would expect.

Proposition 5 *Let π be a protocol that securely computes a functionality f in the presence of malicious adversaries. Then π securely computes f in the presence of* augmented *semi-honest adversaries.*

Proof: Let π be a protocol that securely computes f in the presence of malicious adversaries. Let $\mathcal{A}$ be an augmented semi-honest real adversary and let $\mathcal{S}$ be the simulator for $\mathcal{A}$ that is guaranteed to exist by the security of π (for every malicious $\mathcal{A}$ there exists such an $\mathcal{S}$, and in particular for an augmented semi-honest $\mathcal{A}$). We construct a simulator $\mathcal{S}'$ for the augmented semi-honest setting, by simply having $\mathcal{S}'$ run $\mathcal{S}$. However, in order for this to work, we have to show that $\mathcal{S}'$ can do everything that $\mathcal{S}$ can do. In the malicious ideal model, $\mathcal{S}$ can choose whatever input it wishes for the corrupted party; since $\mathcal{S}'$ is *augmented* semi-honest, it too can modify the input. In addition, $\mathcal{S}$ can cause the honest party to output abort. However, $\mathcal{S}'$ *cannot* do this. Nevertheless, this is not a problem because when $\mathcal{S}$ is the simulator for an augmented semi-honest $\mathcal{A}$ it can cause the honest party to output abort with at most negligible probability. In order to see this, note that when two honest parties run the protocol, neither outputs abort with non-negligible probability. Thus, when an honest party runs together with an augmented semi-honest adversary, it too outputs abort with at most negligible probability. This is due to the fact that the distribution over the messages it receives in both cases is identical (because a semi-honest real adversary follows the protocol instructions just like an honest party). This implies that the simulator for the malicious case, when applied to an augmented semi-honest real adversary, causes an abort with at most negligible probability. Thus, the augmented semi-honest simulator can run the simulator for the malicious case, as required. ■

Given the above, it is our position that the definition of *augmented semi-honest adversaries* is the "right way" of modeling semi-honest behavior.

References

[1] A. Beimel, T. Malkin and S. Micali. The All-or-Nothing Nature of Two-Party Secure Computation. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, p.80-97, August 15-19, 1999.

[2] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[3] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In 19*th STOC,* pages 218–229, 1987. For details see [4].

[4] O. Goldreich. *Foundations of Cryptography: Volume 2 – Basic Applications.* Cambridge University Press, 2004.

[5] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols – Techniques and Constructions.* Springer, 2010.