AN ABSTRACT OF THE THESIS OF

Morgan Shirley for the degree of Master of Science in Computer Science presented on June 7, 2017.

Title: On the Structure of Unconditional UC Hybrid Protocols

Abstract approved:

Mike Rosulek

Two parties, Alice and Bob, hold inputs x and y respectively. They wish to compute a function f of their inputs. In an ideal world, $f(x, y)$ could be calculated by sending the inputs to a trusted third party. In the absence of such a third party, Alice and Bob are required to communicate directly. Alice would like the real-world computation of f to reveal no more about her input x to Bob than he could have deduced from the ideal-world interaction. In addition, Alice would like this guarantee to hold even if Bob cheats at the protocol. Bob would like the computation to have the same properties with regards to security against malicious Alice.

If both parties have unlimited computation power, such a feat is impossible for all but the simplest f. We introduce a trusted third party that can compute for Alice and Bob a different function g. If f can be computed in this modified model, we say that f *reduces* to g.

Some g, which we call *complete*, have a special structural property that allows all f to reduce to g. These reductions are well-studied. Unfortunately, if g does not have this structural property, we do not fully understand reductions to g. This thesis describes our work in characterizing this landscape.

In particular, we show that if f reduces to g by some deterministic protocol or by a randomized protocol with a strict sub-logarithmic bound in the number of communication rounds, then we can shorten these protocols to use only a single call to g. In addition, we give a combinatorial property of f and g that is present if and only if this single-call protocol is possible. We also show an example of f and g where a randomized and potentially super-logarithmic protocol is required for f to reduce to g. This example hints at a direction for future investigation towards the complete characterization of these reductions.

On the Structure of Unconditional UC Hybrid Protocols

by
Morgan Shirley

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 7, 2017
Commencement June 2017

ACKNOWLEDGMENTS

Thanks, Mike, for asking me hard questions and teaching me how to be precise. I've really enjoyed working with you.

Thanks to everyone else who has supported me, and in particular my family. It's nice to be able to get away from work for a while, whether it's on a sight-seeing vacation or just a quiet night out. Having these things to look forward to really kept me going. A special thanks to anyone who sat through me explaining this research to them – I really enjoy this stuff, and it's nice to be able to talk about it with people.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

LIST OF FIGURES

1 Introduction

Consider two parties, Alice and Bob. Alice holds a secret input x and Bob holds a secret input y. Alice and Bob wish to evaluate some two-input function f on x and y *securely*. This means that neither party should learn more about the other party's secret input than they could have inferred from seeing the output $f(x, y)$. This guarantee ought to hold even if one of the parties is acting *maliciously* by deviating from whatever agreed-upon protocol the parties are using. This problem is called *2-party secure function evaluation* (or *SFE*).

In this thesis, we only consider SFE when Alice and Bob are allowed unlimited computational power. That is, every result is information-theoretic in nature. We also only consider functions with finite input ranges.

1.1 Examples of Secure Function Evaluation

The following are two illustrative examples of 2-party secure function evaluation.

1.1.1 The Millionaire Problem

Andrew Yao first introduced the idea of SFE by considering the Millionaire Problem [Yao82]. Alice and Bob are both millionaires, and they want to determine which of them is richer. However, they do not wish to share any further information about their net worth. The *only* piece of information that should be shared is a single bit – 0 if Alice is richer, and 1 if Bob is richer.

In this instance of SFE, Alice's net worth is x, Bob's net worth is y, and f is simply the less-than operator ($<$).

1.1.2 Satellite Collisions

Two countries are spying on each other using surveillance satellites. Neither country wishes the other to know the locations of their satellites for obvious reasons. However, both countries realize that it would be incredibly costly if a satellite from one country was to collide with a satellite from the other.

In this instances of SFE, the locations and trajectories of the countries' satellites are x and y, and f is a function that alerts the countries if a collision is likely to occur.

1.2 Security

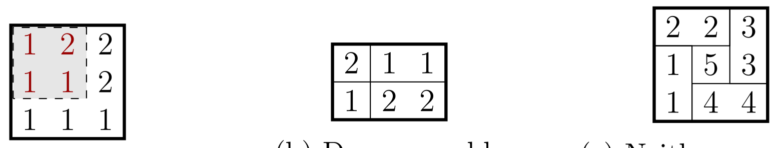
Note that in both of the above examples, each party will learn something about the other party's input. In the Millionaire Problem, if Bob has 7 million dollars and the output reveals that Alice's wealth is less than his, he learns, for example, that Alice does not have 8 million dollars. In the satellite collision problem, finding no collisions reveals that there are certain places where the other country's satellites cannot be.

These pieces of information would have been learned no matter how the function evaluation was performed!

In the *ideal* world, the two parties would have access to a trusted third party. A perfectly secure way of evaluating f would be for Alice to send x to the third party, Bob to send y to the third party, and for the third party to announce $f(x, y)$ publicly. This clearly leaks only information that the parties could learn from examining the output, as no communication actually occurs between Alice and Bob.

In the *real* world, there is no such third party. Instead, Alice and Bob must communicate with each other directly. This interaction is *secure* if anything they can do in the real world could have been done in the ideal world. It turns out that most functions cannot be securely evaluated in the real world without further assumptions! What sorts of assumptions are required to compute a given function in the real world? This is the question that this thesis investigates.

It can be useful to picture functions as matrices, where the rows are inputs in X, the columns are inputs in Y, and the entry in row x and column y is $f(x, y)$. Figure 1 shows three functions in matrix form. Each has some characteristic that affects the assumptions necessary to securely evaluate that function. These properties will be considered in this thesis.



(a) Complete (b) Decomposable (c) Neither complete nor decomposable

Figure 1: Example SFE functions.

1.2.1 Semi-Honest Security and Decomposable Functions

One often-studied restriction on the secure function evaluation model is to force the parties to strictly adhere to a commonly-agreed-upon protocol. They can still try to learn as much as they are able about the other party's input, but they cannot deviate from the set of allowed actions. This model is called *semi-honest*, or *honest-but-curious*.

Beaver & Kushilevitz [Bea89, Kus89] give an exact characterization of secure function evaluation in the semi-honest model: only functions that are *decomposable* are securely computable.

Definition 1. *A function $f : X \times Y \to Z$ is decomposable if either of the following are true:*

- *f is constant*
- *f may be partitioned into two decomposable functions. Either:*
  - *$\exists P, Q$ that partition Y, where $\forall x \in X$, $\forall y \in P$, $\forall y' \in Q$, $f(x, y) \neq f(x, y')$ and $f_P : X \times P \to Z$ and $f_Q : X \times Q \to Z$ are decomposable, or*

– *$\exists P, Q$ that partition X, where $\forall y \in Y$, $\forall x \in P$, $\forall x' \in Q$, $f(x, y) \neq f(x', y)$ and $f_P : P \times Y \to Z$ and $f_Q : Q \times Y \to Z$ are decomposable*

Out of the functions in Figure 1, only Figure 1(b) is decomposable.

An algorithm for securely computing decomposable f in the semi-honest model is as follows:

- If f is constant, return that constant value.
- Otherwise, f may be partitioned along X or Y. If it is partitioned along X, then Alice announces whether she has an input in P or an input in Q. If it is partitioned along Y, Bob announces whether he has an input in P or an input in Q.

This real-world protocol does not leak any information that would not be revealed in the ideal-world protocol. To understand why, we must consider the *rectangles* of decomposable functions.

1.2.2 Rectangles

For an SFE function f, define the following:

$$\mathsf{rect}_f(x, y) = \{x' \mid f(x, y) = f(x', y)\} \times \{y' \mid f(x, y) = f(x, y')\}$$

We refer to $\mathsf{rect}_f(x, y)$ as a **rectangle** of f. Intuitively, $\mathsf{rect}_f(x, y)$ is the largest pair of partitions $X' \subset X$ and $Y' \subset Y$ where $x \in X'$ and $y \in Y'$.

The protocol described above for computing decomposable functions is semi-honestly secure because all it does is narrow down the possibilities of rectangles that may contain the correct output. Since the correct rectangle is already revealed by the output $f(x, y)$, no extra information is leaked by revealing that same information slowly.

Note that different rectangles may have the same value! Figure 2 shows a function with only four output values. However, it has six rectangles. There are two rectangles with value 1, and two rectangles with value 4.

1	1	4	4
2	3	3	3
2	4	1	1

Figure 2: A decomposable function with six rectangles.

An important point is that the parties can distinguish between two rectangles of the same value, and indeed between any two rectangles, in a decomposable function. This is because, in decomposable functions, an input will only contain a single rectangle of a given value. In fact, this property is not unique to decomposable functions. Figure 1(c) is not decomposable but still has rectangles that behave nicely. The fact that Figure 1(a) does *not* have rectangles with this property is what makes it *complete* for *reductions* – these terms will be defined later.

1.2.3 Malicious Security

Decomposability only guarantees semi-honest security. If one party, called the *adversary*, is allowed to arbitrarily deviate from the agreed protocol, then certain events might occur in the real world that would be impossible in the ideal world. This event is called an *attack*. Security under these conditions is called *malicious security*.

One attack against decomposable functions is the *input-switching attack*. The basic idea is that when a party reveals which partition contains their input, they have revealed information about their input without locking in an output. This may allow an adversary to switch its input based on that piece of information. For example, in Figure 2, there is no way in the ideal world for Bob to guarantee that the output is either 2 or 4. However, in the real world, he can simply wait until Alice announces whether she has the top input or the bottom two inputs. If she has the top input, Bob can use one of his rightmost two inputs to force the output to be 4. Otherwise, Bob can use his leftmost input to force the output to be 2.

It turns out that if both parties are allowed unlimited computational power and no further assumptions, only very simple functions are computable with malicious security. In the security model considered in this thesis, UC security, the only functions that can be evaluated with malicious security without further assumptions are those where $f(x, y)$ is constant over either x or y (that is, any change to x or y has no affect on the output) [CKL03].[1]

1.3 Cryptographic Complexity

As we have seen, different functions f have different inherent complexities. Constant functions can be securely realized in the presence of computationally unbounded malicious adversaries, but the function in Figure 1(b) cannot. Similarly, Figure 1(b) can be securely realized in the presence of computationally unbounded, semi-honest adversaries, but the function in Figure 1(c) cannot. In this sense, nondecomposable functions have inherently more "cryptographic complexity" than decomposable functions, which in turn have more complexity than constant functions.

1.3.1 Reductions

Another way to compare the complexity of functions is to use a reduction. We say that f *reduces to* g if there is a secure protocol for f in the presence of a trusted third party that can compute g. The model where calls to ideal g are allowed is called the "g-hybrid model". Analysis of reductions can reveal some very fine-grained complexity distinctions.

This thesis uses a framework called UC security [Can01]. We write $\underline{f \sqsubseteq g}$ to denote that there is a UC-secure protocol securely realizing f against unbounded adversaries

[1]In different models, less restrictive classes of functions are allowed. For example, in standalone security [MPR09] more functions may be computed under malicious security.

in the g-hybrid model. UC security will be defined in Section 2.1.

After defining a notion of reducibility, the most natural step is to identify which objects are *complete* for the reduction. A function g is *complete* (under $\sqsubseteq$) if $f \sqsubseteq g$ for all f. Otherwise we say that g is *incomplete*.

Kilian [Kil88] was the first to consider completeness of SFE functionalities, proving that the oblivious transfer function[2] is complete. Although the result pre-dates the UC model, a later variant of the construction [Kil89] is likely to achieve UC security — *i.e.*, oblivious transfer is complete under the $\sqsubseteq$ reduction that we consider in this work.

Later work characterized exactly which functions are complete (with respect to malicious, unconditional security): for symmetric SFE (where both parties receive the same output) [Kil91], for asymmetric SFE (where only one party receives output) [Kil00], and even for randomized SFE functions [MPR12, KMPS14]. This thesis focuses on symmetric SFE.

1.3.2 Complete Functions

Theorem 1 ([Kil91]). *Let f be a deterministic 2-party SFE. f is complete if and only if* $\exists x, x' \in X,\ y, y' \in Y : f(x,y) = f(x,y') = f(x',y) \neq f(x',y')$.

The proof of this is somewhat involved, and is omitted here.

[2]In *oblivious transfer*, one party has multiple pieces of information. The goal is for the other party to learn some subset of that information. The party that holds the information may not discover what subset is being transferred, and the party that is receiving the information cannot learn anything about any information not in the transferred subset.

One way to visually identify a complete function from its matrix representation is to look for what is called an AND-minor (sometimes an OR-minor). Consider the matrix representations of the AND and OR functions in Figure 3.

$$\boxed{\begin{matrix}0 & 0\\ 0 & 1\end{matrix}}$$

(a) AND

$$\boxed{\begin{matrix}0 & 1\\ 1 & 1\end{matrix}}$$

(b) OR

Figure 3: The AND and OR functions.

These are minimal examples of complete functions. $\text{AND}(0,0) = \text{AND}(0,1) = \text{AND}(1,0) \neq \text{AND}(1,1)$, and $\text{OR}(1,1) = \text{OR}(0,1) = \text{OR}(1,0) \neq \text{OR}(0,0)$. A function contains an AND-minor if you can remove all but two rows and all but two columns from its matrix representation and be left with the AND function (perhaps with some rotation and re-labeling of outputs). Note that a similarly-defined OR-minor is equivalent: OR contains an AND-minor and vice-versa.

The function in Figure 1(a) contains a number of AND-minors, one of which is highlighted. Therefore, it is complete. The other functions in Figure 1 do not contain AND-minors, and therefore are incomplete.

1.3.3 Incomplete Functions

When one or both of f, g are complete, the question of $f \sqsubseteq g$ is simple to answer. If g is complete, then $f \sqsubseteq g$. If f is complete but g is not, then $f \not\sqsubseteq g$.

The goal of this line of work is to therefore **understand when $f \sqsubseteq g$, for f and g which are incomplete.**

Write $f \sqsubset g$ if $f \sqsubseteq g$ but $g \not\sqsubseteq f$. Prabhakaran & Rosulek [PR08] gave an example of four functions that satisfy $f_1 \sqsubset f_2 \sqsubset f_3 \sqsubset f_4$. Maji, Prabhakaran & Rosulek [MPR09] extended this result to show an *infinite* strict hierarchy $f_1 \sqsubset f_2 \sqsubset \cdots \sqsubset f_i \sqsubset \cdots$, and also showed an example of a pair of functions that are incomparable under the reduction: $f \not\sqsubseteq g \wedge g \not\sqsubseteq f$. The same authors in [MPR10] later proved several results of the form "f reduces to g with polynomial-time UC security if and only if one-way functions exist" and "f reduces to g with polynomial-time UC security if and only if semi-honest OT protocols exist". These results imply that $f \not\sqsubseteq g$ (against unbounded adversaries).

These results hint at a rich landscape of complexity with respect to the $\sqsubseteq$ reduction, but fall well short of revealing the entire picture. First, they are not complete characterizations, but give only necessary conditions for $f \sqsubseteq g$. Second, the techniques in these works apply only to f and g that are decomposable. This leaves a gap in our understanding: functions like Figure 1(c) are neither decomposable nor complete.

1.4 Overview of Results

A function f is *unilateral* if there exists an input y^* for one of the parties (by symmetry, Bob) such that $f(\cdot, y^*)$ is a constant function. That is, by choosing input y^* Bob can *unilaterally* fix the output of f.

This thesis shows a complete characterization of when $f \sqsubseteq g$ for a natural class of protocols. This class excludes unilateral f – in Section 6.1 we consider why this exclusion is necessary.

Say that f *embeds in* g if f appears as a submatrix of g, subject to some other restrictions (essentially, the other parts of g can't "interfere" with the f-submatrix — the formal definition is in Chapter 3).

Theorem 2. *The following are equivalent, when f and g are incomplete and f is non-unilateral.*

1. *$f \sqsubseteq g$ via a (worst-case) logarithmic-round protocol*
2. *$f \sqsubseteq g$ via a deterministic protocol*
3. *$f \sqsubseteq g$ via a deterministic protocol consisting of a single call to g and no additional communication.*
4. *f embeds in g*

1.4.1 Technical Approach

The most involved part of the main theorem is proving (1) $\Rightarrow$ (3) and (2) $\Rightarrow$ (3). Intuitively this involves "compressing" an arbitrary protocol for $f \sqsubseteq g$ into a single call to g.

The first step is to show that every secure protocol for $f \sqsubseteq g$ can be transformed into one with the following **instantaneous** property:

- With overwhelming probability, the protocol terminates immediately following some call to g.

- Strictly before this terminal call to g, the protocol transcript leaks negligible information about either party's inputs.

The main technical tool is that of *frontier analysis*, which was introduced in [MPR09] and extended in [MOPR11]. A *frontier* in the protocol is simply the collection of partial transcripts where some statistical condition is true for the first time. Roughly speaking, we define two frontiers for each party: one expressing "the first time the simulator is likely to extract" (the simulator is a feature of the UC-security model, and is defined in Section 2.1) and another for "the first time honest parties can reliably predict the final output." We then argue that these frontiers must all be reached simultaneously, with overwhelming probability. As such, these events can happen only as the result of a call to g. Furthermore, the protocol can be safely truncated after reaching the frontiers (since both parties can already predict the final output). The result of truncation is a protocol with the "instantaneous" property described above.

Next, we show that a protocol with this instantaneous form can be compressed to a single call to g. Whereas the previous step applies for any protocol, for technical reasons we are only able to compress protocols that are deterministic or have logarithmic (worst-case) round complexity.

1.4.2 Tightness of the Characterization

Our main theorem does not apply to super-logarithmic-round, randomized protocols. We show that this limitation is inherent, and that there is actually a *qualitative difference* in the power of super-logarithmic, randomized protocols! In Section 6.2 we demonstrate a specific pair f and g with the following properties:

1. f does not embed in g. Hence, by the classification theorem, $f \not\sqsubseteq g$ via any deterministic protocol or (randomized) logarithmic-round protocol.

2. $f \sqsubseteq g$ via a randomized protocol whose *expected* round complexity is constant, but whose *worst-case* round complexity is $\omega(\log \kappa)$, where κ is an adjustable security parameter of the protocol. Interestingly, this protocol still has the *instantaneous* property described above.

This example paints a truly bizarre picture about the structure of some UC hybrid protocols. We have a secure protocol for f, which leaks no information about the parties' inputs before suddenly revealing all of $f(x, y)$ in a single call to g. Yet any attempt to compress the protocol to just the "meaningful" call to g results in an insecure protocol! Somehow, the fact that the round complexity has the *potential* to be $\omega(\log \kappa)$ seems to be vital for security.

To the best of our knowledge, this is the first example of a round-complexity lower bound in the information-theoretic setting that is not *constant* in the security parameter. Almost all previous work characterizing $f \sqsubseteq g$ focuses on the case where g is complete. In this setting, all protocols can achieve constant rounds: f can be realized in constant rounds from oblivious transfer [IPS08], which itself can be realized in constant rounds from any complete g [KMQ11].[3]

The main theorem also does not characterize $f \sqsubseteq g$ when f is unilateral. This restriction is inherent. In Section 6.1, we demonstrate an example pair of unilateral and incomplete functions f, g where $f \not\sqsubseteq g$ by any protocol consisting entirely of a single call to g. Yet we show a protocol for $f \sqsubseteq g$ consisting of a single call to g plus *one additional message* from one party to the other. Intuitively f has several unilateral inputs, and the extra protocol message is required to convey which one was chosen.

[3]This argument results in a protocol with $O(d_f)$ rounds, where d_f is the multiplicative depth of a circuit realizing f. It is indeed a major open problem to give a protocol with asymptotically fewer rounds. Since in our setting g is incomplete, our goal is not to securely realize *all* f from g. Instead, we consider a fixed f and g, and hence d_f is a constant with respect to the security parameter.

2 Preliminaries

The following is a rigorous definition of 2-party SFE.

A 2-party SFE task is a deterministic function $f : X \times Y \to Z$. We identify f with an *ideal functionality*. This ideal functionality is essentially the "trusted third party" from above. The ideal functionality waits for input $x \in X$ from Alice and input $y \in Y$ from Bob. If no party is adversarial, the functionality gives output $f(x, y)$ to both parties. If any party is adversarial, the functionality gives $f(x, y)$ to the adversary and waits for a command $(\textsc{deliver}, b)$ from the adversary, where $b \in \{0, 1\}$. If $b = 0$, the functionality gives output $\perp$ to the honest party; if $b = 1$, the functionality gives $f(x, y)$ to the honest party. Hence, we consider $\underline{\text{security with abort}}$ — the functionality does not guarantee fairness of output delivery.

In the real world, the parties interact through some protocol π. At each step of the protocol, the parties send some information (or, if allowed, make calls to some ideal functionality g). Either of these things become a part of the *transcript* of the current protocol execution.

2.1 UC Security

The UC security framework was introduced by Canetti [Can01]. An execution in the framework begins with an *environment* $\mathcal{Z}$ (an arbitrary interactive Turing Machine) that chooses inputs for both of the *parties*. The parties interact with each other, and an ideal functionality for some function g, according to a protocol π. The parties eventually give an output to the environment, who outputs a single bit. Throughout the entire interaction, there is an *adversary* $\mathcal{A}$ (an arbitrary interactive TM) who interacts arbitrarily with the environment. The adversary may also choose to *corrupt* one of the parties, which causes the party to come under complete control of the adversary. In that case, the party may deviate from the protocol. In this work we consider only **static** corruption, where the adversary must choose to corrupt a party before the protocol begins.

Figure 4 shows the basic structure of a UC execution.

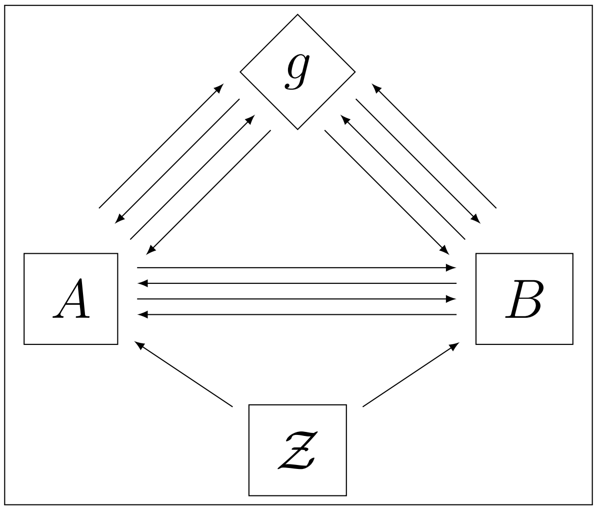


Figure 4: A real-world interaction in the UC framework.

Given the description above, $\textsc{exec}[\pi, g, \mathcal{A}, \mathcal{Z}, 1^\kappa]$ denotes the probability that the environment outputs 1, when κ is the *security parameter* of the protocol.

A particular protocol of interest is the dummy protocol π_{dummy}. In this protocol, each party receives an input from the environment and sends it directly to the ideal functionality. When the ideal functionality delivers an output, the party gives it directly to the environment as output.

Figure 5 shows the structure of π_{dummy}. When f reduces to g, there is a protocol π that uses ideal calls to g (the "real" world) whose behavior (in terms of output distribution and input leakage) looks very similar to π_{dummy} which uses an ideal call to f (the "ideal" world).

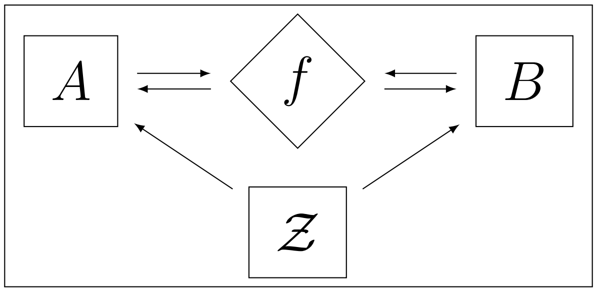


Figure 5: A dummy protocol interaction in the UC framework.

In the ideal world there exists a *simulator* $\mathcal{S}$ that interacts with the corrupted party. The adversary and the simulator carry out protocol π between each other, where $\mathcal{S}$ plays the part of both the honest party and the ideal functionality g. However, the simulator doesn't have access to the honest party or its input! It can only run π_{dummy} with the honest party, with ideal functionality f. Figure 6 shows the structure of this interpretation of an ideal world.

The security arises from the fact that the simulator only communicates with the

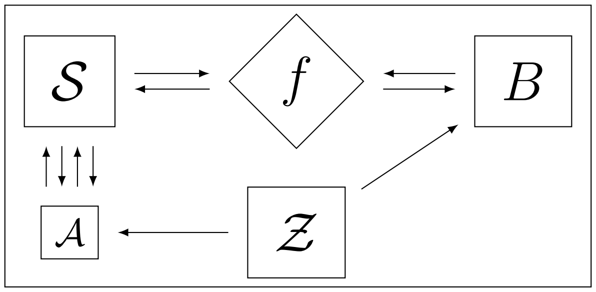


Figure 6: An ideal-world interaction in the UC framework.

honest party via a single call to ideal f. This means that everything in the simulated transcript prior to the call $\mathcal{S}$ makes to ideal f must be independent of the honest party's input, and everything after that call must not affect the honest party's output. We call this event the *extraction*, as $\mathcal{S}$ must "extract" enough information about the input of $\mathcal{A}$ in order to correctly call f.

The formal definition of reduction relies on this interpretation of the ideal and real worlds.

Definition 2. *The* simulation error *of a protocol π is the maximum (supremum) over all environments $\mathcal{Z}$, adversaries $\mathcal{A}$, and simulators $\mathcal{S}$ of the following:*

$$\left| \text{EXEC}[\pi, g, \mathcal{A}, \mathcal{Z}, 1^\kappa] - \text{EXEC}[\pi_{\mathit{dummy}}, f, \mathcal{S}, \mathcal{Z}, 1^\kappa] \right|$$

Definition 3. *We say f reduces to g, and write $f \sqsubseteq g$, if there exists a protocol π that has a simulation error which is negligible in the security parameter κ.*

We write $f \sqsubseteq_1 g$ if furthermore π has the following property: the parties make *only one call to g and exchange no other messages.*

2.2 Properties of g-Hybrid Protocols for Incomplete g

Fix a 2-party protocol π, and let t be a partial transcript (*i.e.*, a prefix of a complete protocol transcript). We use $\underline{\Pr_\pi[t|xy]}$ to denote the probability of obtaining a protocol transcript with prefix t, when both parties run the protocol honestly with respective inputs x and y.

Write t as a sequence of messages $t = (m_1, \dots, m_k)$. Suppose Alice sends the odd-numbered messages. Then the choice of the odd-numbered (resp. even-numbered) messages depends only on the previous messages and x (resp. y), but not on y (resp. x). We can therefore write:

$$\begin{aligned}
\Pr_\pi[t|xy] &= \prod_{i=1}^{k} \Pr_\pi[m_i|xy, m_1 \cdots m_{i-1}] \\
&= \Big(\prod_{i \text{ odd}} \Pr_\pi[m_i|x, m_1 \cdots m_{i-1}] \Big) \Big(\prod_{i \text{ even}} \Pr_\pi[m_i|y, m_1 \cdots m_{i-1}] \Big) \\
&\overset{\text{def}}{=} \Pr_\pi[t|x] \Pr_\pi[t|y] \qquad (\star)
\end{aligned}$$

Here we are defining $\Pr_\pi[t|x]$ and $\Pr_\pi[t|y]$ to be equal to the parenthesized quantities. Essentially, $\Pr_\pi[t|x]$ is the probability that Alice behaves consistently with t when her input is x.

In the g-hybrid model. A similar property also holds when the parties can call an ideal functionality g (*i.e.*, protocols in the g-hybrid model), but only when g is *incomplete*.

When parties invoke g, its output is added to the joint transcript. This is equivalent to adding $\mathsf{rect}_g(\tilde{x}, \tilde{y})$ to the transcript, where $\tilde{x}$ and $\tilde{y}$ were the inputs that the parties gave to this instance of g. Let $\tilde{X} \times \tilde{Y}$ be a particular rectangle in g, then:

$$\Pr[\mathsf{rect}_g(\tilde{x}, \tilde{y}) = \tilde{X} \times \tilde{Y}] = \Pr[\tilde{x} \in \tilde{X}] \Pr[\tilde{y} \in \tilde{Y}]$$

Alice's choice of $\tilde{x}$ depends only on her f-protocol input and the transcript so far; similarly $\tilde{y}$ depends only on Bob's input and the transcript so far. Hence, even protocols in the g-hybrid model satisfy the product property ($\star$).

Stateless parties/adversaries. The "standard" way of defining a protocol is to have each party initially choose a random tape. Then each party's behavior is a deterministic function of the random tape, their input, and the transcript so far.

One way to interpret the product property ($\star$) — for protocols in the g-hybrid model for incomplete g — is that Alice's view (including her private randomness) is independent of Bob's view (including his randomness), given the transcript.

Therefore, *any g-hybrid protocol π* can be purged of stateful randomness in the following way. At each step, a stateless party can (1) sample a random tape conditioned on it being consistent with their private input and transcript so far; (2) use that (ephemeral) random tape to choose the next move in the protocol; (3) discard the ephemeral random tape. Note that this transformation may require exponential time, but we consider all parties to have unbounded computation.

Importantly, this transformation *also applies to adversaries* — that is, without loss of generality we consider only stateless adversaries. The ability to consider only stateless adversaries is perhaps the fundamental property of incomplete g used in this thesis.

3 Reducibility Characterization

We define the combinatorial condition at the heart of our main theorem. Intuitively, f embeds in g if one can identify a submatrix of g that "looks like" f. Of course, the outputs of f might be renamed relative to g. Such a submatrix property suffices for a semi-honest protocol for f using g, where parties simply use the subset of inputs of g that comprise the f-submatrix. However, such a protocol need not be secure in the presence of *malicious* adversaries, because other inputs of g may "interfere" with the f-submatrix. There are two main things that can go wrong. An example of each is shown in Figure 7.

$$f_1 = \boxed{\begin{array}{cc} 1 & 3 \\ 1 & 4 \\ 2 & 4 \end{array}} \not\sqsubseteq \boxed{\begin{array}{ccc} 1 & 3 & 5 \\ 1 & 4 & 6 \\ 2 & 4 & 7 \end{array}} = g_1; \qquad f_2 = \boxed{\begin{array}{cc} 1 & 3 \\ 2 & 4 \end{array}} \not\sqsubseteq \boxed{\begin{array}{ccc} 1 & 3 \rightarrow & 3 \\ 2 & \leftarrow 2 & 4 \end{array}} = g_2$$

Figure 7: Examples of bad embeddings.

Note that f_1 appears as the white submatrix of g_1. But when a corrupt column-player cheats and uses the shaded column of g_1, he completely learns the row-players input. Yet no column of f_1 legally allows this.

Similarly, f_2 appears as a submatrix of g_2. Consider a corrupt column-player who uses the shaded column of g_2. There is no *single* input for f_2 that "explains" the effect of this behavior for all possible inputs of the row-player. Concretely, there is no input of f_2 that guarantees an output in $\{2, 3\}$.

The requirements for embedding are formalized in the following definition:

Definition 4. *For two functions α and β we say that α* ***leaks no more than*** *β if $\beta(y) = \beta(y') \Rightarrow \alpha(y) = \alpha(y')$ for all inputs y, y'. We say that α* ***refines*** *β if $\beta(y) \in \{\alpha(y), \perp\}$ for all inputs y.*

Let $f : X \times Y \to Z$ and $g : \widehat{X} \times \widehat{Y} \to \widehat{Z}$. Without loss of generality, assume $f(x,y) = \mathsf{rect}_f(x,y)$ and $g(x,y) = \mathsf{rect}_g(x,y)$. We say that f ***embeds in*** *g if:*

1. *(f appears as a submatrix in g) There exist two injective mappings, $A : X \to \widehat{X}$ and $B : Y \to \widehat{Y}$, and a third mapping, $C : \widehat{Z} \to Z \cup \{\perp\}$, such that $\forall x \in X, y \in Y : f(x,y) = C(g(A(x), B(y)))$.*

2. *(security guarantees) There exist mappings $\widehat{A} : \widehat{X} \to X$ and $\widehat{B} : \widehat{Y} \to Y$ such that the following hold:*

   (a) (g doesn't reveal too much information)

   - *for all $\widehat{x} \in \widehat{X}$, $g(\widehat{x}, B(\cdot))$ leaks no more than $f(\widehat{A}(\widehat{x}), \cdot)$*
   - *for all $\widehat{y} \in \widehat{Y}$, $g(A(\cdot), \widehat{y})$ leaks no more than $f(\cdot, \widehat{B}(\widehat{y}))$*

   (b) (there are no ambiguous g-inputs)

   - *for all $\widehat{x} \in \widehat{X}$, $f(\widehat{A}(\widehat{x}), \cdot)$ refines $C(g(\widehat{x}, B(\cdot)))$.*
   - *for all $\widehat{y} \in \widehat{Y}$, $f(\cdot, \widehat{B}(\widehat{y}))$ refines $C(g(A(\cdot), \widehat{y}))$.*

To understand this definition, it helps to see how the mappings $A, B, C, \widehat{A}, \widehat{B}$ relate to a secure protocol demonstrating $f \sqsubseteq_1 g$:

Lemma 3. *If f embeds in g, then $f \sqsubseteq_1 g$ via a deterministic protocol. This proves (4) $\Rightarrow$ [(1) $\wedge$ (2) $\wedge$ (3)] of Theorem 2.*

Proof. Let f embed in g, with associated mappings as in Definition 4. The protocol for f is as follows:

- Alice sends input $A(x)$ to g where x is her f-input.
- Bob sends input $B(y)$ to g where y is his f-input.
- The parties both output $C(z)$ where z is the output they receive from g (they output $\perp$ if g gives output $\perp$).

Correctness follows from the first condition of Definition 4. Due to the symmetry in the definitions/protocol, we show security only against a malicious Alice.

Suppose Alice sends input $\widehat{x}$ to g. In the real protocol, Alice's view will consist of $g(\widehat{x}, B(y))$ and Bob's output will be $C(g(\widehat{x}, B(y)))$. In the ideal world, the simulator will do the following:

- The simulator sends $x^* = \widehat{A}(\widehat{x})$ to the ideal f, and obtains output $f(x^*, y)$, which is $C(g(A(x^*), B(y)))$.
- The simulator does not know Bob's input y but can choose any y' such that $f(x^*, y') = f(x^*, y)$. The simulator can give $g(\widehat{x}, B(y'))$ to Alice as her simulated view. From part 2a of Definition 4, we have that this is identical to the real view $g(\widehat{x}, B(y))$.

- The simulator checks whether $C(g(\widehat{x}, B(y))) = \bot$, and if so sends $(\textsc{deliver}, 0)$ to f. In this case, Bob's real and ideal outputs will both be $\bot$. Otherwise, it sends $(\textsc{deliver}, 1)$ to f and Bob will receive output $f(x^*, y)$.

Bob's ideal output is $f(x^*, y) = C(g(A(x^*), B(y)))$. From condition 2b of Definition 4, this matches the real output $C(g(\widehat{x}, B(y)))$. $\square$

Lemma 4. *For non-unilateral f, if $f \sqsubseteq_1 g$ via a deterministic protocol with simulation error less than 1, then f embeds in g. This proves (3) $\Rightarrow$ (4) of Theorem 2.*

Proof. During the deterministic protocol, Alice and Bob both map their f-inputs to g-inputs and then immediately terminate the protocol, meaning that they were each able to map the g-output to the same f-output. The input maps are A and B in the embedding. The output map is C in the embedding.

By symmetry, we only consider security against a malicious Alice.

Clearly, A is injective. If it is not, then choose some pair (x, x') where $A(x) = A(x')$. Choose y so that $f(x, y) \neq f(x', y)$ – since f is not unilateral, such an input exists. Consider two environments – they ask the parties to honestly run the protocol with inputs x, y and x', y respectively, and return 1 if the output is correct. In the real world, in both cases the parties will return the same output, so one of them is incorrect. Therefore, one of the environments has simulation error 1.

Because this protocol is UC-secure, there must be some method by which the simulator takes the parties' g-inputs and translate them to f-inputs upon extraction.

Call these mappings for Alice and Bob $\widehat{A}$ and $\widehat{B}$, respectively. It suffices to show that these mappings satisfy 2a and 2b of Definition 4. We show that if the mappings violate Definition 4, then the simulation error of the protocol is 1.

(2a) Assume that $\widehat{A}$ violates part 2a. That is, there exists an $\widehat{x}$ where it is not the case that $g(\widehat{x}, B(\cdot))$ leaks no more than $f(\widehat{A}(\widehat{x}), \cdot)$. In particular, there is some pair (y, y') where $f(\widehat{A}(\widehat{x}), y) = f(\widehat{A}(\widehat{x}), y')$ but $g(\widehat{x}, B(y)) \neq g(\widehat{x}, B(y'))$.

Consider two environments:

- Adversary Alice uses input $\widehat{x}$, honest Bob uses input y, return 1 if Alice's view is $g(\widehat{x}, B(y))$.
- Adversary Alice uses input $\widehat{x}$, honest Bob uses input y', return 1 if Alice's view is $g(\widehat{x}, B(y'))$.

In the real world, both environments output 1 with probability 1. In the ideal world, the simulator's view in both environments is $f(\widehat{A}(\widehat{x}), y) = f(\widehat{A}(\widehat{x}), y')$. Since the simulator is deterministic, it must give the same simulated g-output for both environments. However, $g(\widehat{x}, B(y)) \neq g(\widehat{x}, B(y'))$, so in at least one of the environments the probability of outputting 1 is 0. Therefore, the simulation error is 1.

(2b) Assume that $\widehat{A}$ violates part 2b. Then there exists some $\widehat{x}$ and y where $C(g(\widehat{x}, B(y))) \notin \{f(\widehat{A}(\widehat{x}), y), \perp\}$. Consider the environment in which corrupt Alice uses g-input $\widehat{x}$ and honest Bob uses input y, and the environment outputs 1 if

Bob's output is $C(g(\widehat{x}, B(y)))$. The environment outputs 1 with probability 1 in the real world. But in the ideal world, Bob's output is either $f(\widehat{A}(\widehat{x}), y)$ or $\perp$. This environment demonstrates a simulation error of 1. $\square$

4 Instantaneous Protocols

In this chapter we show how to transform any secure protocol in the g-hybrid model into one that has an "instantaneous" property (described further in Section 4.4). The results in this section apply to arbitrary protocols. Later in Section 5 we give further transformations that are restricted to deterministic or logarithmic-round protocols.

4.1 Frontier Basics

Recall $\Pr_\pi[\mathcal{E}|xy]$ denotes the probability of event $\mathcal{E}$ when the protocol is run honestly with inputs x and y. Following Section 2.2, we write $\Pr_\pi[\mathcal{E}|xy] = \Pr_\pi[\mathcal{E}|x]\Pr_\pi[\mathcal{E}|y]$. Specifically, if t is a partial protocol transcript, we write $\Pr_\pi[t|xy]$ to refer to the probability of generating a transcript that has t as a prefix.

We write $\Pr_\pi[\mathcal{E}|txy]$ to denote the probability that event $\mathcal{E}$ happens, given that the parties run with inputs x and y, and conditioned on t occurs as a prefix. More formally, the probability considers what happens when the parties run the protocol honestly with inputs x and y, but are initialized with t as the partial transcript.

Let F be any set of partial protocol transcripts, with the property that if $t \in F$, and t is a prefix of t', then $t' \in F$. In other words, F describes an event in the protocol that happens and does not "unhappen." In this case we call F a **frontier**, using the terminology of [MPR09].

It is sometimes helpful to associate the frontier F with its set of prefix-minimal elements, as these represents transcript where *some condition happened for the first time.* Let $\mathsf{first}(F)$ denote the prefix-minimal elements of F.

If F is a frontier, we use notation $\Pr_\pi[F|xy]$ to denote the probability that F is encountered when running the protocol honestly on inputs x and y. More formally:

$$\Pr_\pi[F|xy] \overset{\text{def}}{=} \textstyle\sum_{t \in \mathsf{first}(F)} \Pr_\pi[t|xy]$$

Finally, if F and G are two frontiers, then "$F < G$" denotes the event "either F happens strictly before G, or F happens and G never happens." More formally,

$$\Pr_\pi[F < G|xy] \overset{\text{def}}{=} \textstyle\sum_{t \in \mathsf{first}(F \setminus G)} \Pr_\pi[t|xy]$$

4.2 Our Frontiers

Our analysis relies on two types of frontiers that we introduce:

$F^x_{\textsf{A-ext}}$: captures the first time that the *simulator extracts* with reasonable probability, in an ideal-world interaction involving a corrupt Alice running *honestly* on input x.

$F^x_{\textsf{A-out}}$: captures the first time that Alice's output becomes relatively fixed, in the following sense. If the parties continue with honest behavior from such a point in the protocol, and Alice has input x, then Alice has only one likely output, no matter what Bob's input is.

We define such a frontier for every input x. We also define analogous frontiers with respect to Bob.

We have already defined $\Pr_\pi[\,\cdot\,|xy]$ notation with respect to an honest execution of the protocol on inputs x, y. Since $F^x_{\mathsf{A\text{-}ext}}$ refers to probabilities in an *ideal-model* interaction, we introduce notation to differentiate between the probabilities in real and ideal interactions. We write $\Pr_{\mathsf{A\text{-}sim}}[\,\cdot\,|xy]$ to refer to probabilities induced by an ideal-model interaction among malicious Alice running the protocol honestly on input x, the simulator for corrupt Alice, and ideal honest Bob with input y. $\Pr_{\mathsf{B\text{-}sim}}$ is defined analogously.

While frontiers have been used before to prove lower bounds about UC protocols [MPR09, MOPR11], one novel aspect of our approach is to define a frontier explicitly in terms of the *simulator's behavior.* This choice appears to simplify the technical arguments about frontiers happening in a particular order. Previous work defined frontiers only in terms of the protocol's real-interaction.

Definition 5. *At some point in an ideal interaction between corrupt Alice and the simulator, the simulator will at some point "extract" by sending an input to the ideal f. Define:*

$$\sigma_A(t, x) \overset{\text{def}}{=} \Pr_{\mathsf{A\text{-}sim}}[\textit{simulator has already extracted}|txy]$$

That is, $\sigma_{\mathsf{A}}(t,x)$ is the probability that the simulator has extracted, given that the transcript so far is t. We define σ_{B} analogously.

Note that before the simulator extracts, its view is perfectly independent of y in the ideal interaction. Its decision to extract, and hence the probability $\sigma_{\mathsf{A}}(t,x)$, depends only on x and not on y.

Note that as the transcript evolves, the probability of extraction cannot change as a result of a message sent by Alice. It can only change as a result of a message generated by the simulator, hence an output of g or a simulated Bob-message.

Definition 6. *Given a secure protocol π with security error ϵ, define the following for all inputs x, y:*

$$F^x_{\textsf{A-ext}} = \{t \mid \sigma_A(t,x) > 4\sqrt{\epsilon}\}$$

$$F^y_{\textsf{B-ext}} = \{t \mid \sigma_B(t,y) > 4\sqrt{\epsilon}\}$$

$$F^x_{\textsf{A-out}} = \{t \mid \forall y,y' : f(x,y) \neq f(x,y') \Rightarrow \min \left\{ \begin{array}{c} \Pr_\pi[out\ f(x,y)|txy], \\ \Pr_\pi[out\ f(x,y')|txy'] \end{array} \right\} < 1-\sqrt{\epsilon}\}$$

$$F^y_{\textsf{B-out}} = \{t \mid \forall x,x' : f(x,y) \neq f(x',y) \Rightarrow \min \left\{ \begin{array}{c} \Pr_\pi[out\ f(x,y)|txy], \\ \Pr_\pi[out\ f(x',y)|tx'y] \end{array} \right\} < 1-\sqrt{\epsilon}\}$$

Here $\Pr_\pi[out\ z|txy]$ refers to the probability that honest parties output z when starting the protocol at partial transcript t and running honestly with inputs x and y.

To understand $F^x_{\textsf{A-out}}$, observe that for $t \in F^x_{\textsf{A-out}}$ there is at most one output that can be induced with probability at least $1 - \sqrt{\epsilon}$. It may be the case that no valid output can be induced with this probability, in which case only $\perp$ output is likely from starting point t.

Note that if ϵ is a negligible function of the security parameter, then $\sqrt{\epsilon}$ is a larger function but also still negligible.

4.3 Properties of the Frontiers

We now show that, roughly speaking, all the frontiers that we have defined must occur simultaneously, with overwhelming probability. Note that all lemmas hold with the roles of Alice and Bob reversed.

Lemma 5. *For all x, y:* $\Pr_\pi[F^x_{\textsf{A-ext}} < F^y_{\textsf{B-out}} \mid xy] < 2\sqrt{\epsilon}$.

Proof. Let $\textsf{bad} = \textsf{first}(F^x_{\textsf{A-ext}} \setminus F^y_{\textsf{B-out}})$, whose probability we wish to bound. A partial transcript $t \in \textsf{bad}$ represents a situation where there is reasonable probability that a simulator would have extracted an effective input for Alice ($t \in F^x_{\textsf{A-ext}}$), but in the protocol Alice can still induce two different outputs for Bob, each with good probability ($t \notin F^y_{\textsf{B-out}}$). Intuitively, the simulator has extracted prematurely. This event should be rare.

Consider the following strategy for corrupt Alice and environment:

- Run the protocol with input y for honest Bob, and Alice initially behaving semi-honestly with input x.

- If the protocol transcript avoids bad, then the adversary gives up and the environment outputs 0.
- Otherwise, when the partial transcript reaches $t \in \mathsf{bad}$ for the first time, then the properties of bad guarantee that there are two values x_0, x_1 such that $f(x_0, y) \neq f(x_1, y)$ and $\Pr_\pi[\text{out } f(x_c, y) | t x_c y] \geq 1 - \sqrt{\epsilon}$ for both $c \in \{0, 1\}$.
- The adversary sends x_0, x_1 to the environment, who chooses a random $c \leftarrow \{0, 1\}$.
- The adversary switches strategies to run the protocol honestly with input x_c. The environment outputs 1 if Bob's eventual output is $f(x_c, y)$. Otherwise the environment outputs 0.

Let succ denote the event that the environment outputs 1.

In the real interaction, the environment outputs 1 only when the transcript hits bad and the adversary is successful in forcing Bob's output, which happens with probability at least $1 - \sqrt{\epsilon}$ by the properties of bad. So:

$$\Pr_\pi[\mathsf{succ}] \geq \Pr_\pi[\mathsf{bad} | xy](1 - \sqrt{\epsilon})$$

In the ideal interaction (between the adversary and simulator), Bob's output is now determined differently, as the output of the ideal f. There are two ways the environment outputs 0: (1) when the simulated transcript avoids bad; (2) when the transcript reaches bad but the simulator has already extracted. In the latter case, the environment's choice of c is independent of the simulator's extraction, so with further probability at least 1/2 the honest Bob will not output $f(x_c, y)$. So:

$$\Pr_{\mathsf{A\text{-}sim}}[\mathsf{succ}] \leq (1 - (4\sqrt{\epsilon})/2) \Pr_{\mathsf{A\text{-}sim}}[\mathsf{bad}|xy]$$

From the security of the protocol:

$$|\Pr_{\pi}[\mathsf{succ}] - \Pr_{\mathsf{A\text{-}sim}}[\mathsf{succ}]| < \epsilon$$

$$|\Pr_{\pi}[\mathsf{bad}|xy] - \Pr_{\mathsf{A\text{-}sim}}[\mathsf{bad}|xy]| < \epsilon$$

Hence:

$$\begin{aligned}
\epsilon &> \Pr_{\pi}[\mathsf{succ}] - \Pr_{\mathsf{A\text{-}sim}}[\mathsf{succ}] \\
&\geq \Pr_{\pi}[\mathsf{bad}|xy](1 - \sqrt{\epsilon}) - (1 - 2\sqrt{\epsilon}) \Pr_{\mathsf{A\text{-}sim}}[\mathsf{bad}|xy] \\
&\geq \Pr_{\pi}[\mathsf{bad}|xy](1 - \sqrt{\epsilon}) - (1 - 2\sqrt{\epsilon})(\Pr_{\pi}[\mathsf{bad}|xy] + \epsilon) \\
&= \Pr_{\pi}[\mathsf{bad}|xy]\sqrt{\epsilon} - \epsilon(1 - 2\sqrt{\epsilon}) \\
&> \Pr_{\pi}[\mathsf{bad}|xy]\sqrt{\epsilon} - \epsilon
\end{aligned}$$

Solving for $\Pr_\pi[\mathsf{bad}|xy]$:

$$\Pr_\pi[\mathsf{bad}|xy] < \frac{2\epsilon}{\sqrt{\epsilon}} = 2\sqrt{\epsilon} \qquad \square$$

Next we show that $F^x_{\mathsf{A\text{-}out}}$ is a point at which the honest parties can predict their eventual output.

Definition 7. *Fix x and let $t \in F^x_{\mathsf{A\text{-}out}}$. Then there is at most one value z such that $\exists y : \Pr_\pi[out\ z|xyt] > 1 - \sqrt{\epsilon}$. Let $\mathsf{guess}_\mathsf{A}(t,x)$ denote this value z, and note that the value could be $\perp$. We extend the notation $\mathsf{guess}_\mathsf{A}(t,x) = \perp$ in the case that $t \notin F^x_{\mathsf{A\text{-}out}}$.*

Lemma 6. *For $z \neq \perp$ define $G^z = \{t \mid \mathsf{guess}_\mathsf{A}(t,x) = z\}$. Then for all x, y: $\Pr_\pi[G^{f(x,y)}|xy] > 1 - \epsilon/2$. Intuitively, upon reaching $F^x_{\mathsf{A\text{-}out}}$, Alice can predict her eventual output with error at most $\epsilon/2$.*

Proof. Define $\mathsf{bad} = F^x_{\mathsf{A\text{-}out}} \setminus G^{f(x,y)}$. Intuitively, these are the places in the protocol where $\mathsf{guess}_\mathsf{A}(t,x) \neq f(x,y)$.

From the correctness of the protocol, we have:

$$\begin{aligned}
\epsilon &> \Pr_\pi[\text{output not } f(x,y)|xy] \\
&\geq \sum_{t \in \mathsf{first}(\mathsf{bad})} \Pr_\pi[t|xy] \Pr_\pi[\text{out } \mathsf{guess}_\mathsf{A}(t,x)|txy] \\
&\geq \sum_{t \in \mathsf{first}(\mathsf{bad})} \Pr_\pi[t|xy](1 - \sqrt{\epsilon}) = (1 - \Pr_\pi[G^{f(x,y)}|xy])(1 - \sqrt{\epsilon})
\end{aligned}$$

Solving for the probability expression:

$$\Pr_\pi[G^{f(x,y)}|xy] \geq 1 - \tfrac{\epsilon}{1-\sqrt{\epsilon}} > 1 - \epsilon/2 \qquad \square$$

Lemma 7. *For all x, y, if x is not a unilateral input for f, then* $\Pr_\pi[F^x_{\textsf{\textit{A-out}}} < F^x_{\textsf{\textit{A-ext}}} \mid xy] < 16\epsilon$*.*

Proof. Let $\mathsf{bad} = \mathsf{first}(F^x_{\textsf{A-out}} \setminus F^x_{\textsf{A-ext}})$, whose probability we wish to bound. A partial transcript $t \in \mathsf{bad}$ represents a situation where Alice can predict what the output will be ($t \in F^x_{\textsf{A-out}}$), but the simulator probably has not extracted yet ($t \notin F^x_{\textsf{A-ext}}$). This event should be rare, since in the ideal world Alice can gain no information about the f-output before the simulator extracts.

Let x, y be given in the premise of the lemma. Since x is not a unilateral input, let y' be such that $f(x, y) \neq f(x, y')$. Consider the following interaction with a corrupt Alice and environment:

- Alice initially runs the protocol honestly with input x. The environment randomly chooses input $y^* \leftarrow \{y, y'\}$ for Bob. If the transcript avoids bad then the adversary gives up and the environment outputs 0.
- Otherwise, if the partial transcript reaches $t \in \mathsf{bad}$, there is a unique $z = \mathsf{guess}_{\mathsf{A}}(t, x)$ such that $\Pr_\pi[\text{out } z|tx] > 1 - \sqrt{\epsilon}$. The adversary reports z to the environment.

- The environment outputs 1 if $z = f(x, y^*)$.

Let succ denote the probability that the environment outputs 1.

In the real interaction, the environment outputs 0 only if the transcript avoids bad or if $\mathsf{guess}_{\mathsf{A}}(t, x)$ is incorrect. By the union bound and Lemma 6,

$$\Pr_\pi[\neg\mathsf{succ}] \leq \Pr_\pi[\neg\mathsf{bad}|xy^*] + \Pr_\pi[\neg G^{f(x,y^*)}|xy^*] \leq 1 - \Pr_\pi[\mathsf{bad}|xy^*] + \epsilon/2$$

In the ideal interaction, the environment outputs 0 in the following (mutually exclusive) scenarios: (1) the simulated transcript avoids bad; (2) the transcript reaches bad and the simulator has not yet extracted. In the latter case, the adversary's view is independent of the environment's choice of y^*, and so the environment outputs 0 with further probability at least 1/2. Hence:

$$\begin{aligned}
\Pr_{\mathsf{A\text{-}sim}}[\neg\mathsf{succ}] &\geq \Pr_{\mathsf{A\text{-}sim}}[\neg\mathsf{bad}|xy^*] + \Pr_{\mathsf{A\text{-}sim}}[\mathsf{bad} \wedge \text{no extract}|xy^*]/2 \\
&\geq 1 - \Pr_{\mathsf{A\text{-}sim}}[\mathsf{bad}|xy^*] + \Pr_{\mathsf{A\text{-}sim}}[\mathsf{bad}|xy^*](1 - 4\sqrt{\epsilon})/2 \\
&= 1 - \Pr_{\mathsf{A\text{-}sim}}[\mathsf{bad}|xy^*](\tfrac{1}{2} + 2\sqrt{\epsilon})
\end{aligned}$$

Combining:

$$\begin{aligned}\epsilon &> \Pr_\pi[\mathsf{succ}] - \Pr_{\mathsf{A\text{-}sim}}[\mathsf{succ}] \\ &\geq \Pr_\pi[\mathsf{bad}|xy^*] - \epsilon/2 - \Pr_{\mathsf{A\text{-}sim}}[\mathsf{bad}|xy^*](\tfrac{1}{2} + 2\sqrt{\epsilon}) \\ &\geq \Pr_\pi[\mathsf{bad}|xy^*] - \epsilon/2 - (\Pr_\pi[\mathsf{bad}|xy^*] + \epsilon)(\tfrac{1}{2} + 2\sqrt{\epsilon}) \\ &= \Pr_\pi[\mathsf{bad}|xy^*](\tfrac{1}{2} - 2\sqrt{\epsilon}) - \epsilon(1 - 2\sqrt{\epsilon})\end{aligned}$$

Solving for the probability expression:

$$\Pr_\pi[\mathsf{bad}|xy^*] \leq \frac{\epsilon(2 - 2\sqrt{\epsilon})}{(\frac{1}{2} - 2\sqrt{\epsilon})} < \frac{2\epsilon}{1/4} = 8\epsilon$$

Since $\Pr_\pi[\mathsf{bad}|xy^*]$ is the average of $\Pr_\pi[\mathsf{bad}|xy]$ and $\Pr_\pi[\mathsf{bad}|xy']$, it follows that $\Pr_\pi[\mathsf{bad}|xy] < 16\epsilon$. □

Lemma 8. *For all x, y, if y is not a unilateral input, then* $\Pr_\pi[F^y_{\mathsf{B\text{-}out}} < F^x_{\mathsf{A\text{-}out}}|xy] < 18\sqrt{\epsilon}$.

Proof. By a union bound,

$$\begin{aligned}\Pr_\pi[F^y_{\mathsf{B\text{-}out}} < F^x_{\mathsf{A\text{-}out}}|xy] &\leq \Pr_\pi[F^y_{\mathsf{B\text{-}out}} < F^y_{\mathsf{B\text{-}ext}}|xy] + \Pr_\pi[F^y_{\mathsf{B\text{-}ext}} < F^x_{\mathsf{A\text{-}out}}|xy] \\ &\leq 16\epsilon + 2\sqrt{\epsilon} < 18\sqrt{\epsilon}\end{aligned}$$

□

Lemma 9. *For all x, y, if neither x nor y are a unilateral input, then* $\Pr_\pi[F^y_{\mathsf{B\text{-}out}} <$

$F^x_{\textsf{A-ext}}|xy] < 34\sqrt{\epsilon}$.

Proof. By a union bound,

$$\begin{aligned}
\Pr_\pi[F^y_{\textsf{B-out}} < F^x_{\textsf{A-ext}}|xy] &\leq \Pr_\pi[F^y_{\textsf{B-out}} < F^y_{\textsf{B-ext}}|xy] \\
&+ \Pr_\pi[F^y_{\textsf{B-ext}} < F^x_{\textsf{A-out}}|xy] \\
&+ \Pr_\pi[F^x_{\textsf{A-out}} < F^x_{\textsf{A-ext}}|xy] \\
&\leq 16\epsilon + 2\sqrt{\epsilon} + 16\epsilon < 34\sqrt{\epsilon}
\end{aligned}$$

□

Lemma 10. *Let F be any frontier in the protocol. For all x, y, y',*

$$\left| \Pr_\pi[F < F^x_{\textsf{A-ext}}|xy] - \Pr_\pi[F < F^x_{\textsf{A-ext}}|xy'] \right| < 6\sqrt{\epsilon}$$

Proof. Let $G = F \setminus F^x_{\textsf{A-ext}}$. The main idea is that in the ideal interaction with corrupt Alice, it is unlikely that the simulator has extracted before the protocol has reached G. Conditioned on the simulator not yet extracting, the transcript is completely independent of Bob's input.

Consider running the ideal interaction and halting it when the transcript reaches either F or $F^x_{\textsf{A-ext}}$. Halting at this point is sufficient to determine whether the event $F < F^x_{\textsf{A-ext}}$ happened. We obtain two interactions depending on whether Bob is given input y or y'. In the terminology of Bellare-Rogaway [BR06], these are two *identical-until-bad games*, where the "bad" event is that the simulator extracts but $F^x_{\textsf{A-ext}}$ is not immediately reached. By the definition of $F^{x}_{\textsf{A-ext}}$, the bad event happens with probability at most $4\sqrt{\epsilon}$. This probability of the bad event bounds the distinguishing bias between the two games.

Then applying the security of the protocol we have:

$$\begin{aligned}\Big|\Pr_{\pi}[F < F^{x}_{\textsf{A-ext}}|xy] - \Pr_{\pi}[F < F^{x}_{\textsf{A-ext}}|xy']\Big| \\ &\leq \Big|\Pr_{\textsf{A-sim}}[F < F^{x}_{\textsf{A-ext}}|xy] - \Pr_{\textsf{A-sim}}[F < F^{x}_{\textsf{A-ext}}|xy']\Big| + 2\epsilon \\ &< 4\sqrt{\epsilon} + 2\epsilon < 6\sqrt{\epsilon}\end{aligned}$$

□

Lemma 11. *For all x, y', y, none of them unilateral,* $\Pr_{\pi}[F^{y'}_{\textsf{B-out}} < F^{y}_{\textsf{B-out}}|xy] < 42\sqrt{\epsilon}$.

Proof. Let $\mathsf{bad} = \mathsf{first}(F^{y'}_{\textsf{B-out}} \setminus F^{y}_{\textsf{B-out}})$, whose probability we wish to bound. We partition bad into two parts: $\mathsf{bad}_1 = \mathsf{bad} \cap F^{x}_{\textsf{A-ext}}$ and $\mathsf{bad}_2 = \mathsf{bad} \setminus F^{x}_{\textsf{A-ext}}$.

Since $\mathsf{bad}_1 \subseteq F^{x}_{\textsf{A-ext}} \setminus F^{y}_{\textsf{B-out}}$ (i.e., the event $F^{x}_{\textsf{A-ext}} < F^{y}_{\textsf{B-out}}$ is true for these transcripts), Lemma 5 implies that

$$\Pr_{\pi}[\mathsf{bad}_1|xy] < 2\sqrt{\epsilon}.$$

Since bad_2 happens strictly before the $F^{x}_{\textsf{A-ext}}$ event, Lemma 10 implies that

$$\Big|\Pr_{\pi}[\mathsf{bad}_2|xy] - \Pr_{\pi}[\mathsf{bad}_2|xy']\Big| < 6\sqrt{\epsilon}.$$

Since $\mathsf{bad}_2 \subseteq F^{y'}_{\textsf{B-out}} \setminus F^{x}_{\textsf{A-ext}}$, Lemma 9 implies that

$$\Pr_\pi[\mathsf{bad}_2|xy'] < 34\sqrt{\epsilon}.$$

Putting everything together, we have:

$$\begin{aligned}\Pr_\pi[\mathsf{bad}|xy] &\leq \Pr_\pi[\mathsf{bad}_1|xy] + \Pr_\pi[\mathsf{bad}_2|xy] \\ &< 2\sqrt{\epsilon} + \Pr_\pi[\mathsf{bad}_2|xy'] + 6\sqrt{\epsilon} \\ &< 2\sqrt{\epsilon} + 34\sqrt{\epsilon} + 6\sqrt{\epsilon} \\ &= 42\sqrt{\epsilon} \qquad \square\end{aligned}$$

4.4 Securely Truncating a Protocol

Lemma 12. *Let π be a secure protocol for f in the g-hybrid model. Define π' to be the following:*

- *On input x for Alice and y for Bob, both parties run π honestly on their given inputs.*
- *When the protocol transcript t reaches $F^{\tilde{x}}_{\textsf{A-out}}$ for any $\tilde{x}$, or reaches $F^{\tilde{y}}_{\textsf{B-out}}$ for any $\tilde{y}$, the parties terminate the protocol.*
- *Alice outputs $\mathsf{guess}_A(t, x)$ and Bob outputs $\mathsf{guess}_B(t, y)$.*

Then the truncated protocol π' is also a secure protocol for f.

Proof. Let ϵ denote the simulation error of π. First, we argue that π' is correct. Alice's output is $\mathsf{guess}_{\mathsf{A}}(t,x)$, which differs from the correct answer $f(x,y)$ only in the following events:

- $t \notin F^{x}_{\mathsf{A\text{-}out}}$ because the protocol reached reached $F^{x'}_{\mathsf{A\text{-}out}}$ and terminated strictly before reaching $F^{x}_{\mathsf{A\text{-}out}}$ for $x' \neq x$. By Lemma 8, this can happen only with probability $O(\sqrt{\epsilon})$.

- $t \notin F^{x}_{\mathsf{A\text{-}out}}$ because the protocol reached reached $F^{y}_{\mathsf{B\text{-}out}}$ and terminated strictly before reaching $F^{x}_{\mathsf{A\text{-}out}}$. By Lemma 11, this can happen only with probability $O(\sqrt{\epsilon})$.

- $t \in F^{x}_{\mathsf{A\text{-}out}}$ but $\mathsf{guess}_{\mathsf{A}}(t,x) \neq f(x,y)$. By Lemma 6, this can only happen with probability $O(\epsilon)$.

As for security, the only difference between π and π' is that π' truncates early based on some condition. But this condition is public and *independent of either party's private inputs.* Hence the simulation for π' works as follows. It simply runs the simulator for π but terminates the protocol when the transcript reaches the public termination condition.

Overall π' is a secure protocol with negligible simulation error $O(\sqrt{\epsilon})$. □

Observe that the new protocol π' has the "instantaneous" property discussed in Section 1.4. Importantly for our purposes in the next section, with overwhelming probability $1 - O(\sqrt{\epsilon})$ the protocol terminates on a transcript that is both in $F^x_{\textsf{A-out}}$ and $F^y_{\textsf{B-out}}$. Such a transcript must end with a message produced by the simulator in both ideal interactions (i.e., when either party is corrupt). Hence the last protocol message must be an output of g, with overwhelming probability.

5 Collapsing Protocols to a Single Call to g

We complete our main theorem with the following lemmas.

In order to collapse a protocol to a single round, we use two important properties of instantaneous protocols. First, by Lemma 12 we can consider only protocols that end with a call to g. Second, by Lemma 10 before the final call to g the parties' inputs do not have a noticeable effect on the distribution of transcripts.

Lemma 13. *For all f and g there is a constant $c_{f,g}$ such that if $f \sqsubseteq_1 g$ via a protocol π with simulation error ε, then $f \sqsubseteq_1 g$ via a* deterministic *protocol π' with simulation error at most $c_{f,g}\varepsilon$.*

Proof. Since π consists of only one call to g, the only choices Alice, Bob, and the simulator can make in the protocol are:

- The mapping of Alice's f-input to her g-input
- The mapping of Bob's f-input to his g-input
- The mappings of either party's g-input to a suitable f-input in the simulator
- The mapping of the g-output to an f-output

The only ways that randomness can manifest in the protocol are in the choice of these mappings.

Let $c_{f,g}$ be the number of possible combinations of such mappings based on these random coins. This is certainly a constant, although it is perhaps very large.

Select the mapping combination that was most likely to be chosen in π. Consider the deterministic protocol π' constructed by locking in these choices at the start of the protocol. The probability that Alice, Bob, and the simulator in π match the behavior of π' is at least $1/c_{f,g}$. Then, if the simulation error of π' is δ, the simulation error of π must be at least $\delta/c_{f,g}$. Therefore, if π has simulation error ϵ, then π' must have simulation error at most $c_{f,g}\varepsilon$. □

Given a protocol π with a strict upper limit of r rounds, $\mathsf{trunc}(\pi, i)$ is the protocol constructed by truncating π after $r - i$ rounds, outputting $\perp$ if π was not finished. Note that $\mathsf{trunc}(\pi, 0) = \pi$.

Let $\mathcal{R}$ be the transcripts of $\mathsf{trunc}(\pi, i)$ which are $r - i - 1$ rounds (that is, there is one action to go in the protocol) but have not terminated yet.

Lemma 14. *If π has simulation error ε, then for all x, y, y' and all i:*

$$\left| \Pr_{\mathsf{trunc}(\pi,i)}[\mathcal{R}|xy] - \Pr_{\mathsf{trunc}(\pi,i)}[\mathcal{R}|xy'] \right| < 6\sqrt{\epsilon}$$

Proof. By Lemma 10 we know that this is true in $\mathsf{trunc}(\pi, 0) = \pi$, as these $\mathcal{R}$ transcripts are strictly before $F^x_{\mathsf{A\text{-}ext}}$. Truncating doesn't change simulator extraction probabilities, as the simulator for $\mathsf{trunc}(\pi, i)$ just runs the simulator for π up until the truncated transcript. Therefore, the lemma still holds with respect to $\mathsf{trunc}(\pi, i)$. □

Lemma 15. *If a protocol π is not ε-secure against malicious Alice, then there is an environment* ENV *for π with the following properties:*

1. ENV *chooses inputs for Bob uniformly at random*

2. $\Pr_\pi[\mathsf{ENV} \text{ outputs } 1] > \frac{1}{2} + \frac{\varepsilon}{4|Y|}$.

3. $\Pr_{\textsf{\textit{A-sim}}}[\mathsf{ENV} \text{ outputs } 1] < \frac{1}{2} - \frac{\varepsilon}{4|Y|}$

Of course, a symmetrical lemma holds for protocol π that is insecure against malicious Bob.

Proof. Take an environment ENV_0 for which π has simulation error ε. Construct ENV as follows.

- Choose Bob's input y^* uniformly at random.

- Let $p(y)$ be the probability that ENV_0 chooses y. Let p_{max} be the maximum $p(y)$.

- Flip a coin with probability $1 - (p(y^*)/p_{\mathsf{max}})$. If it comes up heads, abort and return 0.

- Otherwise, run ENV_0.

Note that $\mathsf{ENV} \equiv \mathsf{ENV}_0$ conditioned on ENV not aborting. We abort with probability at most $(|Y|-1)/|Y|$ (as we never abort for the y where $p(y) = p_{\mathsf{max}}$). $(|Y|-1)/|Y|$ is a constant. The simulation error of ENV is therefore at least $\varepsilon/|Y|$.

At this point, possibly invert the output of ENV such that $\Pr_\pi[\mathsf{ENV}\text{ outputs }1]$ is greater than $\Pr_{\mathsf{A\text{-}sim}}[\mathsf{ENV}\text{ outputs }1]$. This will not affect the simulation error.

The probability of ENV returning 1 in the real or simulated environments differs by $\varepsilon/|Y|$, and is centered around some constant p. That is, the probabilities are at least $p+\delta$ and at most $p-\delta$ respectively, where $\delta = \varepsilon/(2|Y|)$. Perform the following operations to ensure that the probability is centered around $1/2$ as required by the lemma.

1. If $p > 1/2$, run normally with probability $\frac{1}{2p}$. Otherwise, return 0.
2. if $p < 1/2$, run normally with probability $\frac{1}{2(1-p)}$. Otherwise, return 1.

This will "normalize" the average of the probabilities in the real and ideal world to $1/2$. This might shrink δ slightly – up to a factor of 2. The minimum value of δ is $\frac{\varepsilon}{4|Y|}$, as required by the lemma. □

Define $\varepsilon_0 = \sqrt{\varepsilon}$ and $\varepsilon_i = (52nc)\varepsilon_{i-1} = (52nc)^i\varepsilon_0$ where $c = c_{f,g}$ is the constant defined in Lemma 13 and n is the maximum of $|X|$ and $|Y|$.

Lemma 16. *If* $\mathsf{trunc}(\pi, i)$ *has simulation error at most* ε_i *then either* f *embeds in* g *or* $\mathsf{trunc}(\pi, i+1)$ *has simulation error at most* ε_{i+1}.

Proof. Consider any partial transcript t where the next action in $\mathsf{trunc}(\pi, i)$ is for the parties to make a call to g. Let protocol π_t be defined as follows: the parties "fast-forward" to t by imagining the transcript up to that round. They then complete the call to g and exit immediately afterwards.

If there is any π_t with simulation error less than $1/c$ (call such a t *good*), f embeds in g: by Lemma 13, there exists a single-round deterministic protocol for f in a g-hybrid world with simulation error less than 1. Then, by Lemma 4, f embeds in g.

Assume that there is no good t in round $r-i-1$ of $\mathsf{trunc}(\pi, i)$ (that is, in $\mathcal{R}$). We wish to bound the probability $\Pr_{\mathsf{trunc}(\pi,i)}[\mathcal{R}|xy]$ for all x, y in this case. Note that some $t \in \mathcal{R}$ may have simulation error when run against one malicious party but not the other. Let $\mathcal{R}_A$ be those t which have unacceptable simulation error against Alice, and let $\mathcal{R}_B$ be similarly defined for Bob. Then:

$$\Pr_{\mathsf{trunc}(\pi,i)}[\mathcal{R}|xy] \leq \Pr_{\mathsf{trunc}(\pi,i)}[\mathcal{R}_A|xy] + \Pr_{\mathsf{trunc}(\pi,i)}[\mathcal{R}_B|xy]$$

Consider, then, the probability $\Pr_{\mathsf{trunc}(\pi,i)}[\mathcal{R}_A|xy]$. A symmetric argument will work for $\mathcal{R}_B$, and therefore we can use these to get a bound on the overall probability $\Pr_{\mathsf{trunc}(\pi,i)}[\mathcal{R}|xy]$.

For each t in $\mathcal{R}$, consider π_t. The simulation error is at least $1/c$, so by Lemma 15 we can construct an environment ENV_t satisfying:

1. ENV_t chooses inputs for Bob uniformly at random

2. $\Pr_{\pi_t}[\mathsf{ENV}_t \text{ outputs } 1] > \frac{1}{2} + \frac{1}{4|Y|c}$

3. $\Pr_{\mathsf{A\text{-}sim\text{-}for\text{-}}\pi_t}[\mathsf{ENV}_t \text{ outputs } 1] < \frac{1}{2} - \frac{1}{4|Y|c}$

Fix a particular x and y. Consider the following attack:

- Alice runs on x, and the environment chooses input y^* for Bob uniformly at random.
- If the parties reach $t \in \mathcal{R}$, let ENV_t sample an input for Bob. If it samples y^*, run ENV_t, possibly allowing Alice to change her input maliciously. Otherwise, output 0.

The probability that during this attack we output 1 is the probability that we reached $\mathcal{R}$ times the probability that ENV_t chooses input y^* times the probability that ENV_t succeeds at its attack given y^*. These probabilities are independent.

$$\sum_{t \in \mathcal{R}_A} \sum_{y^*} \Pr_{\mathsf{trunc}(\pi,i)}[t|xy^*] \frac{1}{|Y|} \Pr_{\pi_t}[\mathsf{ENV}_t \text{ returns } 1|y^*]$$

If we eliminate the summations, we get the following expression:

$$\left(\Pr_{\mathsf{trunc}(\pi,i)}[\mathcal{R}_A|xy^*] \right) \left(\frac{1}{2} + \frac{1}{4|Y|c} \right)$$

By Lemma 14 we can replace y^* with the input for Bob we desire to bound probability against, y, with a small change in probability of reaching a transcript.

The probability the environment outputs 1 in the real world is therefore at least:

$$\left(\Pr_{\mathsf{trunc}(\pi,i)}[\mathcal{R}_A|xy] - 6\sqrt{\varepsilon} \right) \left(\frac{1}{2} + \frac{1}{4|Y|c} \right)$$

By a similar series of arguments, the probability the environment outputs 1 in the ideal world is at most:

$$\left(\Pr_{\textsf{A-sim-for-trunc}(\pi,i)}[\mathcal{R}_A|xy] + 6\sqrt{\varepsilon}\right)\left(\frac{1}{2} - \frac{1}{4|Y|c}\right)$$

Because the simulation error of $\textsf{trunc}(\pi, i)$ is ε_i, the above expression is at most the following value:

$$\left(\Pr_{\textsf{trunc}(\pi,i)}[\mathcal{R}_A|xy] + 6\sqrt{\varepsilon} + \varepsilon_i\right)\left(\frac{1}{2} - \frac{1}{4|Y|c}\right)$$

We know that, because $\textsf{trunc}(\pi, i)$ has a simulation error of ε_i, the difference between the output probabilities in the real and ideal worlds is at most ε_i, which means that, in particular:

$$\begin{aligned}
\varepsilon_i &\geq \Pr_{\textsf{trunc}(\pi,i)}[\mathcal{R}_A|xy]\frac{1}{2|Y|c} - 12\sqrt{\varepsilon} - \varepsilon_i\left(\frac{1}{2} - \frac{1}{4|Y|c}\right) \\
&\Rightarrow \Pr_{\textsf{trunc}(\pi,i)}[\mathcal{R}_A|xy] \leq \left(|Y|c + \frac{1}{2}\right)\varepsilon_i + 24|Y|c\sqrt{\varepsilon} \\
&\Rightarrow \Pr_{\textsf{trunc}(\pi,i)}[\mathcal{R}_A|xy] \leq (26|Y|c)\,\varepsilon_i
\end{aligned}$$

Recall that this is only against malicious Alice. We get the following bound without restricting which party is adversarial:

$$\Pr_{\mathsf{trunc}(\pi,i)}[\mathcal{R}|xy] \leq (52|Y|c)\,\varepsilon_i = \varepsilon_{i+1}$$

Truncating directly before $\mathcal{R}$, then, will only increase the simulation error to ε_{i+1}. □

Lemma 17. *If, for incomplete and non-unilateral f and g, $f \sqsubseteq g$ via a protocol with strict upper bound on number of rounds $r = O(\log\kappa)$, then f embeds in g. This proves (1) ⇒ (4) of Theorem 2 (stated in Section 1.4).*

Proof. $\mathsf{trunc}(\pi, 0)$ has simulation error ϵ which is surely less than ϵ_0.

Either f embeds in g or we can apply the argument in Lemma 16 up to $r - 1$ times. If $r = O(\log\kappa)$, then we are left with a 1-round protocol $\mathsf{trunc}(\pi, r-1)$ with simulation error $\epsilon_r = (52nc)^{O(\log\kappa)}\sqrt{\varepsilon} = \mathsf{poly}(\kappa)\kappa^{-\omega(1)} = \kappa^{-\omega(1)}$ which is negligible. Then $f \sqsubseteq_1 g$, which by Lemma 4 means that f actually does embed in g. □

Corollary 18. *If $f \sqsubseteq g$ via a deterministic protocol (of any number of rounds) then f embeds in g. This proves (2) ⇒ (4) of Theorem 2 (stated in Section 1.4).*

Proof. Deterministic protocols have zero simulation error (without loss of generality). Therefore, the same reasoning as in the previous proof applies but without any error accumulating with each round. □

6 Tightness of the Characterization

In this section we discuss why our main characterization does not extend (without modification) to consider unilateral functions or superlogarithmic-round, randomized protocols.

6.1 Unilateral Functions

In Figure 8 we give f and g which are unilateral (similar examples can be constructed in which g is non-unilateral). Bob is the column-player and thus has 2 unilateral inputs labeled B and C.

First, we argue that $f \not\sqsubseteq_1 g$. Suppose for sake of contradiction that such a protocol exists. Consider the simulator for a corrupt Bob who chooses his f-input uniformly at random and runs the protocol semi-honestly. The only message that the simulator sees is Bob's input to g, after which the simulator must extract an output to send to f. The simulator gets only one bit of information about Bob's input, while there are 3 possibilities for it to send to f. It follows that with constant probability the simulator must extract the wrong input, and this error will be evident in the output of f.

However, there is a simple protocol for f using g: Alice sends her f input directly to g. If Bob has f-input A, he should choose g-input A'. In this case, the parties will see that the g-output is in $\{0, 1\}$ and they terminate with this as their f-output. Otherwise, if Bob has f-input B or C, he should choose g-input B'. In this case, the parties will see that the g output is 2, and then Alice will wait for Bob to send a plain message containing either “2” or “3.” Alice takes this message to be her output.

It is simple to see that this protocol is secure against a malicious Alice. For a malicious Bob, the simulator does the following. If Bob chooses g-input A', then the simulator extracts Bob's ideal f-input as A and simulates the g-output to equal the ideal f-output. If Bob chooses g-input B', then the simulator gives 2 as the simulated g-output, then waits for a message from Bob (either “2” or “3”) and uses this as the extracted ideal f-input. The reason the simulation is secure is that in the second case (Bob chooses g-input B'), the fact that this is a unilateral input means that the simulator doesn't need to know Alice's input to perfectly simulate the g-output. Hence the simulator can delay extraction until the second protocol message, where intuitively Bob resolves which unilateral input he has.

Hence, we have $f \sqsubseteq g$ via a protocol consisting of a single call to g, plus (in some cases) one extra message. It is a deterministic, constant-round protocol, and yet $f \not\sqsubseteq_1 g$. This example shows that our classification does not extend to unilateral functions.

$$
f = \begin{array}{ccc} A & B & C \\ \hline 0 & 2 & 3 \\ 1 & 2 & 3 \end{array}
\qquad
g = \begin{array}{cc} A' & B' \\ \hline 0 & 2 \\ 1 & 2 \end{array}
$$

Figure 8: Unilateral functions violating the main theorem.

$$
f = \begin{array}{|ccc|} \hline \mathrm{A} & \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} & \mathrm{D} \\ \mathrm{C} & \mathrm{E} & \mathrm{E} \\ \hline \end{array}
\qquad
g = \begin{array}{|ccc:ccc:ccc|}
\hline
\mathbf{A} & 1 & 2 & \mathbf{A} & 1 & 2 & \mathbf{B} & 1 & 2 \\
3 & \mathbf{A} & 4 & 3 & \mathbf{A} & 4 & 4 & \mathbf{B} & 3 \\
6 & 5 & \mathbf{A} & 6 & 5 & \mathbf{A} & 5 & 6 & \mathbf{B} \\
\hdashline
\mathbf{C} & 1 & 2 & \mathbf{D} & 1 & 2 & \mathbf{D} & 1 & 2 \\
3 & \mathbf{C} & 4 & 3 & \mathbf{D} & 4 & 4 & \mathbf{D} & 3 \\
6 & 5 & \mathbf{C} & 6 & 5 & \mathbf{D} & 5 & 6 & \mathbf{D} \\
\hdashline
\mathbf{C} & 1 & 4 & \mathbf{E} & 1 & 4 & 4 & 1 & \mathbf{E} \\
6 & \mathbf{C} & 2 & 6 & \mathbf{E} & 2 & \mathbf{E} & 6 & 2 \\
3 & 5 & \mathbf{C} & 3 & 5 & \mathbf{E} & 5 & \mathbf{E} & 3 \\
\hline
\end{array}
$$

Figure 9: Functions violating the main theorem via a superlogarithmic-round protocol.

6.2 Deterministic / Logarithmic-Round Protocols

Consider the functions f and g in Figure 9. We first claim that f does not embed in g. Any embedding would map 3 f-columns into 3 distinct g-columns.[1] For any 3 columns of g, there exists a row for which these columns have distinct entries – this is simple (albeit time-consuming) to verify. However, there is no row in f that has three distinct values. Hence the embedding would contradict rule 2a of the embedding definition. Concretely, any candidate protocol for $f \sqsubseteq_1 g$ would allow a corrupt row-player to learn the column-player's input in its entirety, which is not allowed by f.

However, there is a protocol for f that uses g. We group the rows and columns of g into groups of three, as distinguished by the dotted lines in the figure. Associate the first row of f with the first row group of g, etc. Similarly, associate the first column of f with the first column group of g, etc. The protocol for f is as follows:

[1]Perhaps columns are mapped to rows if the roles of Alice and Bob are swapped during the embedding. The analysis is the same for this scenario.

- Alice chooses a g-input from the row group associated with her f-input, uniformly at random.
- Bob chooses a g-input from the column group associated with his f-input, uniformly at random.
- They call g with their selected g-inputs.
- If the output of the g-call was in {A, B, C, D, E}, terminate the protocol with that output. Otherwise, repeat (with fresh random choices for the g-inputs).

The correctness of this protocol is clear. By only sending g-inputs in the group associated with their f-inputs, each party restricts any terminating output of g to be one that was possible given their f-input.

To see that the protocol is secure, consider the following simulation. Suppose corrupt Alice chooses some g-input (row). With probability 1/3, the simulator decides that the protocol will terminate at this round. It converts the g-input to an f-input (according to its row group), sends that f-input to the ideal f, then simulates the g-output as the ideal f-input. With probability 2/3, the simulator decides that the protocol will continue. Note that in any row, there are 2 non-terminal g-outputs (for example, in the second row only 3 and 4 are possible), which are equally likely no matter which column group Bob has selected. The simulator simply chooses one of these two with equal probability as the simulated g-output. Then the same process repeats.

The parties' inputs will "match" by giving a terminal output with probability 1/3, meaning that the expected number of rounds is 3. The probability that the protocol continues for at least r rounds is $(2/3)^r$. We can get a protocol with a strict upper bound on round complexity by having the parties simply abort after some limit r number of rounds. If we set this limit as $r(\kappa) = \omega(\log \kappa)$, then the correctness of the protocol suffers by an amount $(2/3)^{\omega(\log \kappa)} = \kappa^{-\omega(1)}$, which is negligible. However, the simulation is still perfect, and the protocol is secure.

6.3 Round Complexity and Parallel Calls to g

Our model encompasses protocols that make only a single call to g in each round. Requiring sequential calls to g is without loss of generality *with respect to security*, since in the UC model it cannot be *guaranteed* that calls happen in parallel. The adversary can without loss of generality schedule all the calls sequentially (learning the output of one before choosing an input to the next), resulting in a protocol in our model.

However, when considering round complexity, it is more realistic to allow protocols that make *parallel calls* to g. Although the adversary can schedule these parallel calls in sequence, we still consider the round complexity as that required by the *honest* parties. Most of our technical results apply to such protocols. More formally, consider protocols where at each step the parties may call n parallel instances of g, where n is agreed-upon by both parties. All of our results in Section 4.3 and most of the results in Section 5 apply to such protocols. In particular, we can collapse any $f \sqsubseteq g$ protocol of $O(\log \kappa)$ rounds to a single-round protocol that may make *many* parallel calls to g but uses no additional communication.

To extend our results, it suffices to show that such a protocol (many parallel calls to g, no additional communication) implies that f embeds in g. We have been currently unable to extend this step. The way we currently extract an embedding from a *single-call* protocol (Lemma 4) works by derandomizing the protocol, crucially using the fact that the number of possible actions in the protocol is *constant*. This property is not true when a protocol makes, say, $O(\kappa)$ parallel calls to g.[2]

In this section we gave a $\omega(\log \kappa)$-round protocol for specific f using specific g. We point out that this particular protocol *cannot* be made constant-round by making all the g-calls in parallel. The simple attack is to split the g-calls into two groups and use different effective inputs in both (e.g., Alice uses inputs from the first row-group in half of the calls, and second row-group in the other half). With very good probability, this attack leaks as much as evaluating f on two inputs. In particular, Alice can learn Bob's input in its entirety from this attack.

We conjecture that there is no $O(\log \kappa)$-round protocol for this f using this g, even when the protocol allows unlimited parallel calls to g in each round.

[2]Our results hold as stated for protocols that call at most $O(\log \kappa)$ instances of g in parallel at a time, where the number of possible actions is polynomial in κ.

7 Conclusion

When we first began the research for this thesis, we hoped to fully characterize when $f \sqsubseteq g$ for 2-party secure function evaluations. While we were unfortunately unable to do this, we did manage to fully describe a significant subset of function reductions. Furthermore, the counterexamples we discovered during this process will help direct further research in this area.

For example, it seems likely that there is some clever way to extend our embedding result to include unilateral functions. Define $\underline{\mathsf{trim}(f)}$ to be the result of removing all but one of the unilateral inputs for each party. Then let us write $\underline{f \sqsubseteq_{1+} g}$ when there exists a secure protocol for f that uses one call to g and at most one additional message. We conjecture that, if f is unilateral, then $f \sqsubseteq_{1+} g$ if and only if $\mathsf{trim}(f)$ embeds in g.

The realm of randomized, worst-case super-logarithmic protocols seems like a more formidable obstacle to a full characterization. We were able to generate a number of function pairs with the same form as the example in Figure 9. Frustratingly, for some of these pairs, f actually did embed in g, meaning $f \sqsubseteq_1 g$! Future work down this path might include determining when the randomized protocol is necessary. Another interesting question is whether or not there are other forms of f and g for which $f \not\sqsubseteq_1 g$ but $f \sqsubseteq g$ by a randomized protocol.

Bibliography

[Bea89] Donald Beaver. Perfect privacy for two-party protocols. In Joan Feigenbaum and Michael Merritt, editors, *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography*, volume 2, pages 65–77. American Mathematical Society, 1989.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 68–86. Springer, Heidelberg, May 2003.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

[Kil89] Joe Kilian. *Uses of Randomness in Algorithms and Protocols.* PhD thesis, Department of Electrical Engineering and Computer Science, Mas sachusetts Institute of Technology, 1989.

[Kil91] Joe Kilian. A general completeness theorem for two-party games. In *23rd ACM STOC*, pages 553–560. ACM Press, May 1991.

[Kil00] Joe Kilian. More general completeness theorems for secure two-party computation. In *32nd ACM STOC*, pages 316–324. ACM Press, May 2000.

[KMPS14] Daniel Kraschewski, Hemanta K. Maji, Manoj Prabhakaran, and Amit Sahai. A full characterization of completeness for two-party randomized function evaluation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 659–676. Springer, Heidelberg, May 2014.

[KMQ11] Daniel Kraschewski and Jörn Müller-Quade. Completeness theorems with constructive proofs for finite deterministic 2-party functions. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 364–381. Springer, Heidelberg, March 2011.

[Kus89] Eyal Kushilevitz. Privacy and communication complexity. In *30th FOCS*, pages 416–421. IEEE Computer Society Press, October / November 1989.

[MOPR11] Hemanta K. Maji, Pichayoot Ouppaphan, Manoj Prabhakaran, and Mike Rosulek. Exploring the limits of common coins using frontier analysis of protocols. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 486–503. Springer, Heidelberg, March 2011.

[MPR09] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Complexity of multi-party computation problems: The case of 2-party symmetric secure function evaluation. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 256–273. Springer, Heidelberg, March 2009.

[MPR10] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Cryptographic complexity classes and computational intractability assumptions. In Andrew Chi-Chih Yao, editor, *ICS 2010*, pages 266–289. Tsinghua University Press, January 2010.

[MPR12] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. A unified characterization of completeness and triviality for secure function evaluation. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 40–59. Springer, Heidelberg, December 2012.

[PR08] Manoj Prabhakaran and Mike Rosulek. Cryptographic complexity of multi-party computation problems: Classifications and separations. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 262–279. Springer, Heidelberg, August 2008.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.