Completeness Theorems with Constructive Proofs for Symmetric, Asymmetric and General 2-Party-Functions

Manuscript

Daniel Kraschewski
Jörn Müller-Quade

April 11, 2008
Abstract

In this paper by surprisingly simple criteria a full characterization is given of all complete 2-party-functions, i.e. we give easily checkable criteria, such that for every deterministic and stateless 2-party-primitive with finite input and output alphabets can be decided, whether it allows for implementation of oblivious transfer in an unconditionally secure way. Hence some older results of another author (Joe Kilian 1991, 2000) are unified and generalized to primitives that are neither symmetric nor asymmetric. The proof in this paper is fully constructive and contains protocols that in our view can be understood more easily than the older ones.

We prove security of our protocols in a universally composable setting, but we want to emphasize that our classification results quite directly carry over to every reasonable notion of security we know of.

Contents

1 Introduction 1

2 Presentation of our results 1
   2.1 Notion of security 2
   2.2 Considered primitives 2
   2.3 Basic definitions and notations 3
   2.4 The classification theorem 5

3 Proof of the Classification Theorem 6
   3.1 Secure generation of correlated data 8
      3.1.1 Stochastic basics 15
      3.1.2 General properties of attacks on offline-protocols 19
      3.1.3 Resistant offline-protocols 24
   3.2 Reduction of OT to resistant offline-protocols 29
      3.2.1 The reduction protocol 29
      3.2.2 Correctness of the reduction 35
      3.2.3 Security against a malicious Alice 35
      3.2.4 Security against a malicious Bob 37
      3.2.5 Passive adversaries 39

4 Conclusion 39
1 Introduction

Oblivious transfer (OT) in the sense of a trusted erasure channel was introduced in [1] and later in [2] proven to be equivalent to \( (\binom{2}{1}) \)-OT, where a receiver Bob may learn only one of two bits sent by Alice. OT turned out to be complete in the sense that every secure multiparty computation can be implemented using OT [3, 4, 5]. In order to classify further complete 2-party-functions, in [6, 7] criteria were found for completeness of symmetric (both parties always receive the same output) and asymmetric 2-party-functions (only one party receives some output at all). Noisy classical channels also have been investigated and shown to be complete [8] as well as quantum channels in combination with an additional classical bit commitment\(^1\) [11, 12]. Similar results are known for bounded-classical-storage models [13] and bounded-quantum-storage models [14] respectively. In this paper we now generalize some classical classification results by closing the gap between [6] and [7].

The lightly misguiding naming of symmetric and asymmetric 2-party-functions disguises the existence of a third class (both parties receive different outputs), which we call general 2-party-functions or just 2-party-functions, since symmetric and asymmetric 2-party-functions can be seen as specializations of this class of cryptographic primitives. In this paper by use of surprisingly simple criteria we give a full characterization of all 2-party-functions that allow for implementation of oblivious transfer statistically secure against computationally unlimited active or passive adversaries\(^2\). We give constructive proofs for all cases and our protocols are all running in polynomial time, i.e. honest parties do not need to be more powerful than generally assumed in cryptography.

2 Presentation of our results

In this section we just present our results, since they can be formulated with remarkably little effort. The proofs are given in Section 3. Before we define

---

\(^1\)For quantum channels alone there is a no-go theorem, which is based on the impossibility of unconditional secure quantum bit commitment [9, 10]. Without additional assumptions a quantum channel does not allow for secure implementation of OT.

\(^2\)According to the type of adversary there are slightly different criteria, of course. Interestingly we can use exactly the same protocol scheme in both cases. Passive adversaries just allow for use of a greater class of 2-party-functions.
the class of primitives considered in this paper, let us start with some words about the notion of security that we want to use.

2.1 Notion of security

Basically we use the notion of security of [15]. Nonetheless our results are expected to hold for every reasonable notion of security. In particular we do not use any specific property of that one of [15] until the end of Section 3.1, but prove some statistical lemma instead that holds for every pair of machines running our protocol. With Corollary 24 from Section 3.1.3 and the considerations from Section 3.2 it should be not too hard to find a security proof for our protocol with respect to any reasonable notion of security.

For the sake of completeness we want to give a brief review about the notion of security of [15] here. It is a simulation based notion of security, i.e. an ideal model is compared to a real model. The protocol of interest is running in the latter, where an adversary \( A \) coordinates the behaviour of all corrupted parties. In the ideal model all parties are connected solely by an ideal functionality \( F \), which is an \( (\frac{1}{2}) \)-OT in our case. The corrupted parties in the ideal model are controlled by a simulator \( S \). The protocol inputs of all parties are chosen by an environment \( Z \). During the protocol run this environment \( Z \) may interact with the adversary \( A \) or the simulator \( S \) respectively. In the end \( Z \) learns the protocol outputs of all parties and then it outputs a bit \( z \), that indicates a guess whether this was a run in the real model or in the ideal model. The protocol is said to be secure, if no environment \( Z \) can distinguish between the real and the ideal model better than with some negligible advantage. In particular a protocol \( \pi \) is called a secure implementation of an ideal functionality \( F \), iff for every adversary \( A \) there exists a simulator \( S \), such that for every environment \( Z \) the value \(|\Pr[z = 1 | \pi, A, Z, k] - \Pr[z = 1 | F, S, Z, k]|\) is negligible in the security parameter \( k \). For a security proof it always suffices to consider the so-called “dummy adversary” only, that gives direct control over all corrupted parties to the environment.

2.2 Considered primitives

In this paper we consider the class of all deterministic and stateless 2-party-primitives with finite input alphabets and that are independent of the actual
security parameter. These primitives we call 2-party-functions\textsuperscript{3}.

Notation. For a given 2-party-function \( F \) we denote the input alphabets of Alice and Bob by \( \Omega_A(F) \) and \( \Omega_B(F) \) respectively or just \( \Omega_A \) and \( \Omega_B \) for short. For some subsets of input symbols \( S \subseteq \Omega_A(F) \) and \( T \subseteq \Omega_B(F) \) we denote by \( F(S,T) \) the corresponding 2-party-function with these reduced input alphabets. When \( F \) is invoked with input \( x \in \Omega_A(F) \) from Alice and \( y \in \Omega_B(F) \) from Bob, we denote Alice’s output by \( F_A(x,y) \) and Bob’s output by \( F_B(x,y) \).

We use this notation in that way, that always a fixed 2-party-function \( F \) with input alphabets \( \Omega_A \) and \( \Omega_B \) is given. W.l.o.g. we always implicitly assume \( \Omega_A = \{0, \ldots, |\Omega_A| - 1\} \subseteq \mathbb{N} \) as well as \( \Omega_B = \{0, \ldots, |\Omega_B| - 1\} \subseteq \mathbb{N} \). If there is a second 2-party-function \( G \), we denote its input alphabets explicitly by \( \Omega_A(G) \) and \( \Omega_B(G) \), whereas by \( \Omega_A \) and \( \Omega_B \) always the input alphabets of \( F \) are meant.

This notation of 2-party-functions leads to a canonical representation by matrizes like the following\textsuperscript{4}:

\[
F_A = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \quad F_B = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 2 \end{pmatrix}
\]

When \( F \) is invoked with input \( x \in \Omega_A \) from Alice and \( y \in \Omega_B \) from Bob, Alice’s output is the entry of row \( x \) and column \( y \) of \( F_A \), while Bob’s output is the entry of row \( x \) and column \( y \) of \( F_B \).

2.3 Basic definitions and notations

The representation of 2-party-functions by matrizes as seen in Section 2.2 is quite convenient but a bit too detailed, since simple renaming of input or output symbols leads to different representations. We want to abstract from such unimportant details by canonical equivalence relations.

Definition 1 (Consistent renaming). Let \( F \) and \( G \) be two arbitrary 2-party-functions. We call \( G \) a consistent renaming of \( F \), if \( F \) can be transformed into \( G \) by (repeated) application of the following operations:

\textsuperscript{3}Such primitives in other papers also have been referred to as “crypto gates”.

\textsuperscript{4}This example is the smallest complete 2-party-function that is neither symmetric nor asymmetric.
• permutation of the order of rows of $F$, i.e. renaming Alice’s inputs

• permutation of the order of columns of $F$, i.e. renaming Bob’s inputs

• application of an injective function to all entries of an arbitrary but fixed row of $F_A$, i.e. renaming exactly that part of Alice’s output that corresponds to one specified input symbol of her

• application of an injective function to all entries of an arbitrary but fixed column of $F_B$, i.e. renaming exactly that part of Bob’s output that corresponds to one specified input symbol of him

It is irrelevant, in which order these operations are applied.

Obviously within a cryptographic protocol every invocation of a 2-party-function $F$ can be replaced canonically by an invocation of a consistent renaming of $F$ without any side effects.

If there is only a passive adversary, Definition 1 provides everything we need. For active adversaries larger equivalence classes are needed. In particular we want to get rid of input symbols that an “intelligent” active adversary would never use. We mark such symbols as “redundant”.

**Definition 2** (Redundancy). Let $F$ be an arbitrary 2-party-function. An input symbol $y' \in \Omega_B$ and the corresponding column of $F$ is redundant, iff there is another input symbol $y \in \Omega_B\{y'\}$ with the following two properties:

\[
\forall x \in \Omega_A : \quad F_A(x, y') = F_A(x, y)
\]
\[
\forall x, x' \in \Omega_A : \quad F_B(x, y') \neq F_B(x', y') \Rightarrow F_B(x, y) \neq F_B(x', y)
\]

In this case $y$ is dominated by $y'$.

Redundant rows of $F$ are defined analogously. A 2-party-function without any redundant rows and columns is called redundancy-free.

In Definition 2 a situation is described, where Bob always can input $y$ instead of $y'$ without changing Alice’s output nor losing any information he would have gathered by inputting $y'$. Hence in presence of an active adversary w.l.o.g. we may restrict our considerations to redundancy-free 2-party-functions. The following definition provides the corresponding equivalence relation.
Definition 3. Two arbitrary 2-party-functions $F$ and $G$ are equivalent, iff $F$ can be transformed into a consistent renaming of $G$ by iterated adding and/or removal of redundant rows and/or columns one after another\textsuperscript{5}. The equivalence class of $F$ with respect to this relation is denoted by $[F]$.

By use of Defintion 1 and Definition 3 we can abstract from all irrelevant details of the matrix representation of 2-party-functions. An easily checkable completeness criterion is still missing. Such a criterion can be formulated by use of the following special class of 2-party-functions.

Definition 4 (Minimal-OT). Let $F$ be a 2-party-function with

$$F_A, F_B \in \left\{ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \right\}, \text{ but not } F_A = F_B = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$ 

Every consistent renaming of such a 2-party-function $F$ we call a minimal-OT\textsuperscript{6}.

These four definitions are all we need to present our classification theorem, what is the main result of this paper.

2.4 The classification theorem

With the definitions from Section 2.3 we now can formulate easily checkable but nonetheless quite comprehensive criteria for the structure of complete 2-party-functions.

Theorem 5 (Classification Theorem). Let $F$ be an arbitrary 2-party-function and let $G$ be a redundancy-free 2-party-function with $[G] = [F]$. Then holds:

1. With respect to passive security $F$ is complete, if and only if $F$ contains a minimal-OT as a submatrix.

2. If $F$ is not complete with respect to passive security, then $F$ is a consistent renaming of a symmetric 2-party-function.

\textsuperscript{5}This step-by-step procedure is crucial, since otherwise one could “duplicate” a 2-party-function and then delete it completely.

\textsuperscript{6}If $F$ is a minimal-OT and one of the matrices $F_A$ and $F_B$ is the all-zero matrix, then by invocation of $F$ with uniformly distributed random input Alice and Bob can perform a classical Rabin-OT. If $F$ is symmetric, some more advanced protocol is needed.
3. With respect to active security $F$ is complete, if and only if $G$ contains a minimal-OT as a submatrix.

4. If $F$ is not complete with respect to active security, then $G$ is a consistent renaming of a symmetric 2-party-function.

3 Proof of the Classification Theorem

Instead of proving Theorem 5 directly, we consider the following Theorem 6, which implicates the assertions of Theorem 5.

Theorem 6.

1. Each equivalence class of 2-party-functions as defined in Definition 3 contains either only complete 2-party-functions or only non-complete 2-party-functions with respect to active security.

2. Every 2-party-function that does not contain a minimal-OT is a consistent renaming of a symmetric 2-party-function.

3. Every symmetric 2-party-function that does not contain a minimal-OT is neither complete with respect to active nor passive security.

4. Every redundancy-free 2-party-function that contains a minimal-OT is complete with respect to active security.

5. Every 2-party-function that contains a minimal-OT is complete with respect to passive security.

Proof. The complete proof of Theorem 6 is really complex. Thus we do not give it in full length here, but split it up into several lemmata and just list the links here.

1. The proof of the first assertion of Theorem 6 is almost trivial and hence ommitted here.

2. The second assertion is proven with Lemma 7 right below.

3. For the third assertion it suffices to prove, that every symmetric 2-party-function that does not contain a minimal-OT is not complete with respect to passive security. Such a proof is given in [6], where only
symmetric 2-party-functions are considered and a symmetric minimal-OT is called an "imbedded OR". Although another notion of security is used there, the arguments directly carry over.

4. The proof of the fourth assertion is quite complex and the main part of this paper. Section 3.1 and Section 3.2 solely consist of this proof.

5. The proof of the fifth assertion can be derived easily from the proof of the fourth assertion. This is further explained in Section 3.2.5.

Lemma 7 (Symmetrization-Lemma). Every 2-party-function that does not contain a minimal-OT is a consistent renaming of a symmetric 2-party-function.

Proof. Let \( F \) be a 2-party-function that does not contain a minimal-OT. We want to find a symmetric 2-party-function \( H \) that is a consistent renaming of \( F \). For this purpose we first define a consistent renaming \( G \) of \( F \):

\[
G_A(x, y) := \{ y' \in \Omega_B \mid F_A(x, y) = F_A(x, y') \}
\]

\[
G_B(x, y) := \{ x' \in \Omega_A \mid F_B(x, y) = F_B(x', y) \}
\]

Obviously Alice’s new output alphabet is the power set of Bob’s input alphabet and vice versa. Now starting from \( G \) we define the symmetric 2-party-function \( H \):

\[
H_A(x, y) = H_B(x, y) := (G_A(x, y), G_B(x, y))
\]

We now have to show, that \( H \) is a consistent renaming of \( G \). It suffices to prove the following two equivalences:

\[
\forall x \in \Omega_A, y, y' \in \Omega_B : \quad G_A(x, y) = G_A(x, y') \iff H_A(x, y) = H_A(x, y')
\]

\[
\forall x, x' \in \Omega_A, y \in \Omega_B : \quad G_B(x, y) = G_B(x', y) \iff H_B(x, y) = H_B(x', y)
\]

By definition of \( H \) this is equivalent to the following two implications:

\[
\forall x \in \Omega_A, y, y' \in \Omega_B : \quad G_A(x, y) = G_A(x, y') \implies G_B(x, y) = G_B(x, y')
\]

\[
\forall x, x' \in \Omega_A, y \in \Omega_B : \quad G_B(x, y) = G_B(x', y) \implies G_A(x, y) = G_A(x', y)
\]

Since both of these assertions can be shown analogously, it suffices to prove the first one. We want to give a proof by contradiction and hence assume, that we can find some \( x \in \Omega_A \) and \( y, y' \in \Omega_B \) with:

\[
G_A(x, y) = G_A(x, y') \quad (1)
\]

\[
G_B(x, y) \neq G_B(x, y') \quad (2)
\]
Because of (2) we further find some \( x' \in \Omega_A \) with:
\[
x' \not\in G_B(x, y) \iff x' \in G_B(x, y')
\]
By construction of \( G \) especially holds:
\[
\begin{align*}
x' & \in G_B(x, y) \iff F_B(x', y) = F_B(x, y) \\
x' & \in G_B(x, y') \iff F_B(x', y') = F_B(x, y')
\end{align*}
\]
Hence follows:
\[
F_B(x', y) \neq F_B(x, y) \iff F_B(x', y') = F_B(x, y')
\]
Furthermore because of (1) we must have by construction of \( G \):
\[
F_A(x, y) = F_A(x, y')
\]
Altogether now follows, that \( F(\{x, x'\}, \{y, y'\}) \) is a minimal-OT, no matter if \( F_A(x', y) = F_A(x', y') \) holds or not. Hence we have the needed contradiction, what concludes our proof.

3.1 Secure generation of correlated data

Our reduction from OT to an arbitrary complete 2-party-function is split up into two steps. First Alice and Bob generate some amount of correlated data by repeated invocation of \( F \) with randomized input\(^7\). Within a testphase each party has to partially unveil its data, so that significant cheating can be detected. In a second step on top of the untested correlated data an invocation of OT is built. Within Section 3.1 we consider the first step, the other one is topic of Section 3.2.

Before we can formulate a protocol-scheme for generation of correlated data, we need some adequate notations for handling strings.

**Notation.** Let \( s \) be a finite string over an alphabet \( \Omega \). By \( s[i] \) we denote the \( i \)-th symbol in \( s \). For a given index set \( K = \{k_1, k_2, \ldots, k_n\} \subset \mathbb{N} \) we denote the string \( s[k_1]s[k_2] \ldots s[k_n] \) by \( s[K] \). By \( |s| \) we denote the length of \( s \). By \( |s|_\alpha \)

\(^7\)As far as we know, this is the standard approach of every protocol that implements OT on top of some other primitive \( P \). Since the underlying primitive \( P \) is invoked in this first step only, the main OT computation can be done “offline”, i.e. \( P \) does not have to be accessible during the whole time of a protocol run.
with \( \alpha \in \Omega \) we denote the number of appeareances of \( \alpha \) in \( s \). We canonically extend this notation to subalphabets \( T \subseteq \Omega \) by \( |s|_T := \sum_{\alpha \in T} |s|_{\alpha} \). Further for some given strings \( s_A \) and \( s_B \) of the same length \( |s_A| = |s_B| \) we define the compound string \( s_A \times s_B \), whose \( i \)-th entry is just the tuple \((s_A[i], s_B[i])\).

**Definition 8** (Offline-protocol). An offline-protocol\(^8\) for a 2-party-function \( F \) is characterised by the following parameters, which are all dependent on the security parameter \( k \):

- two distribution functions \( n_A^{(k)} : \Omega_A \to (0, 1) \) and \( n_B^{(k)} : \Omega_B \to (0, 1) \)
- a control parameter \( c^{(k)} \in (0, \frac{k}{2}) \cap \mathbb{Z} \)
- a tolerancy parameter \( \delta^{(k)} \in \mathbb{R}_{>0} \)

For a given security parameter \( k \) the protocol itself proceeds as follows:

1. **Initialization:** Alice initializes two empty strings \( s_A^{\text{in}} \) and \( s_A^{\text{out}} \) as well as an index set \( K_A \), that is initialised by \( K_A := \{1, \ldots, k\} \). Further she initialises an empty set \( K'_A \). Analogously Bob initializes \( s_B^{\text{in}} \), \( s_B^{\text{out}} \), \( K_B \) and \( K'_B \).

2. **Invocation of \( F \):** According to the distribution functions \( n_A^{(k)} \) and \( n_B^{(k)} \) each party randomly chooses an input symbol \( x \in \Omega_A \) and \( y \in \Omega_B \) respectively, then \( F \) is invoked with this input tuple \((x, y)\). Alice learns \( F_A(x, y) \) and concatenates \( x \) to \( s_A^{\text{in}} \) as well as \( F_A(x, y) \) to \( s_A^{\text{out}} \), while Bob learns \( F_B(x, y) \), concatenates \( y \) to \( s_B^{\text{in}} \) and \( F_B(x, y) \) to \( s_B^{\text{out}} \). This protocol step is executed for \( k \) times.

3. **Control A:** Alice randomly chooses an index set \( K'_A \subseteq K_A \) with \( |K'_A| = c^{(k)} \) and sends it to Bob, whose response consists of the strings \( s_B^{\text{in}}[K'_A] := s_B^{\text{in}}[K_A'] \) and \( s_B^{\text{out}}[K'_A] := s_B^{\text{out}}[K_A'] \). Alice aborts the protocol in the following two cases:

- Bob obviously lies, i.e. there is an index \( i \in K_A' \) with:

\[
(s_A^{\text{out}}[i], s_B^{\text{out}}[i]) \neq (F_A(s_A^{\text{in}}[i], s_B^{\text{in}}[i]), F_B(s_A^{\text{in}}[i], s_B^{\text{in}}[i]))
\]

\(^8\)These protocols allow for all invocations of \( F \) to be performed in advance. Hence all the interesting parts of the protocol can be done “offline”, i.e. without access to \( F \).
• Bob’s input distribution significantly differs from its expected value, i.e. there is an input tuple \((x, y) \in \Omega_A \times \Omega_B\) with:

\[
\left| s_{in}^{in}[K'_A] \times s_{in}^{in}[K'_B]|_{(x,y)} - c^{(k)} \cdot n_A^{(k)}(x) \cdot n_B^{(k)}(y) \right| > \delta^{(k)}
\]

At the end of this protocol step all indices from \(K'_A\) are deleted in \(K_A\) and \(K_B\).

4. **Control B:** This protocol step proceeds analogously to **Control A** with interchanged roles of Alice and Bob.

5. **Output:** Alice outputs the compound string \(s_{in}^{in}[K_A] \times s_{out}^{out}[K_A]\) and Bob outputs the compound string \(s_{in}^{in}[K_B] \times s_{out}^{out}[K_B]\).

We are interested in offline-protocols, such that even if one party is corrupted, the output distribution differs from the expected output distribution of a totally uncorrupted setting only by some small polynomial. This is described more formally in Definition 22 in Section 3.1.3. If an offline-protocol meets this demand, then it is sufficiently secure for an implementation of OT.

It is not hard to see that offline-protocols in general are not secure in this sense, e.g. the following example of a 2-party-function \(F\) in fact is complete, but in case of uniform input distributions a corrupted Bob can perfectly simulate honest behaviour without ever using one of the two last input symbols:

\[
F_A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad F_B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}
\]

In this example we can reduce Bob’s cheating possibilities by choosing a less symmetric input distribution for him. If he is forbidden to use the first and the last input symbol at all, a malicious Bob can infringe that only with some subpolynomial frequency or he is caught cheating almost for sure. By the next definition we formulate some kind of asymmetry in input distributions, that is crucial for our security proof.

**Notation.** Let a finite set \(\Omega\) and a function \(g : \Omega \to \mathbb{R}\) be given. For \(T \subseteq \Omega\) we define \(g(T) := \sum_{\omega \in T} g(\omega)\). This especially means \(g(\emptyset) = 0\).

**Definition 9.** Let a 2-party-function \(F\), some non-empty input sets \(S \subseteq \Omega_A\), \(T \subseteq \Omega_B\) and some \(\varepsilon > 0\) be given. We define \(\Pi_F(S, T)\) to be the set of all
offline-protocols for \( F \) with parameters \( n_A, n_B, c \) and \( \delta \), such that for every security parameter \( k \) holds:

\[
\begin{align*}
n^{(k)}_A(S) &= 1 - |\Omega_A \setminus S| \cdot k^{-\frac{1-\epsilon}{8}} \\
\forall x \in \Omega_A : \quad n^{(k)}_A(x) &\geq k^{-\frac{1-\epsilon}{8}} \\
n^{(k)}_B(T) &= 1 - |\Omega_B \setminus T| \cdot k^{-\frac{1-\epsilon}{8}} \\
\forall y \in \Omega_B : \quad n^{(k)}_A(y) &\geq k^{-\frac{1-\epsilon}{8}} \\
c^{(k)} &= \left\lceil k^{\frac{3}{4}+\epsilon} \right\rceil \\
\delta^{(k)} &= k^{\frac{1-\epsilon}{\pi}}
\end{align*}
\]

Although it is formulated more generally, the input sets \( S \) and \( T \) in Definition 9 are meant to be the input sets \( \Omega_A(G) \) and \( \Omega_B(G) \) of some minimal-OT \( G \) in \( F \). Now the question remains, which minimal-OTs do allow for sufficiently secure offline-protocols \( \Pi^2_F(\Omega_A(G), \Omega_B(G)) \). As a warning we first want to state an example \( F \), where the minimal-OT in the upper left corner does not:

\[
F_A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad F_B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 2 & 2 \end{pmatrix}
\]

Nontheless this 2-party-function is complete, e.g. \( G := F(\{0,1\}, \{1,2\}) \) does well. For identification of the “good” minimal-OTs within 2-party-functions in general we need another two definitions.

**Notation.** For a given 2-party-function \( F \) and some input symbols \( x \in \Omega_A, y \in \Omega_B \) by the symbol \((x, y)_F\) a situation is meant, where Alice and Bob invoked \( F \) with input \( x \) and \( y \) respectively.

**Definition 10.** Let a 2-party-function \( F \) and some input symbols \( x \in \Omega_A \) and \( y, y' \in \Omega_B \) be given. For brevity we write \((x, y)_F \sim (x, y')_F\), if for every \( x' \in \Omega_A \) with \( F_B(x, y) = F_B(x', y) \) holds:

\[
F_A(x', y) = F_A(x', y') \quad \land \quad F_B(x, y') = F_B(x', y')
\]

The \( \sim \)-relation describes some kind of local redundancy in the sense that an input symbol \( y' \in \Omega_B \) of a 2-party-function \( F \) is redundant, iff there is another symbol \( y \in \Omega_B \setminus \{y'\} \), such that for every \( x \in \Omega_A \) holds
\((x, y')_F \leadsto (x, y)_F\). In other words Bob can pretend a situation \((x, y)_F\) to be \((x, y')_F\) without any risk of being caught cheating immediately, if and only if \((x, y)_F \leadsto (x, y')_F\).

**Definition 11.** For a given 2-party-function \(F\) and some \(T \subseteq \Omega_B\) we define:

\[
U_B(T) := \{ y \in \Omega_B \mid \exists x \in \Omega_A \ \forall y' \in T : (x, y)_F \nleftrightarrow (x, y')_F \}
\]

\[
V_B(T) := \{ y \in \Omega_B \mid \forall x \in \Omega_A \ \exists y' \in T : (x, y)_F \leadsto (x, y')_F \}
\]

For a given submatrix \(G\) of \(F\) we set \(U_B(G) := U_B(\Omega_B(G))\) as well as \(V_B(G) := V_B(\Omega_B(G))\) for convenience. For Alice’s input sets we define \(U_A\) and \(V_A\) analogously, what also requires a corresponding analogon of the \(\leadsto\)-relation, of course\(^9\).

Intuitively seen \(U_B(T)\) is the set of all inputs \(y \in \Omega_B\) that Bob can not use too often during the run of an offline-protocol \(\pi \in \Pi^\varepsilon_F(\Omega_A, T)\), since there is some \(x \in \Omega_A\) that prevents simulation of honest behaviour in case of a too large number of situations \((x, y)_F\). Thus it seems to be a good idea to look for a minimal-OT \(G\) with \(|U_A(G)|\) and \(|U_B(G)|\) as great as possible. In fact for a minimal-OT \(G\) within a redundancy-free 2-party-function \(F\) one can show that offline-protocols \(\Pi^\varepsilon_F(\Omega_A(G), \Omega_B(G))\) with some small enough \(\varepsilon > 0\) are sufficiently secure, if there is no other minimal-OT \(\hat{G}\) in \(F(\Omega_A, \Omega_B(G))\) with \(V_A(G) \neq V_A(\hat{G}) \subset V_A(G)\) and no minimal-OT \(\hat{G}\) in \(F(\Omega_A(G), \Omega_B)\) with \(V_B(G) \neq V_B(\hat{G}) \subset V_B(G)\). The formal proof of this fact does not need really advanced mathematical concepts, but because of its enormous length we postpone it to Section 3.1.3 (with preliminaries in Section 3.1.1 and Section 3.1.2) and give only a short and very informal sketch of its main idea here. Throughout the whole proof sketch we assume Alice to be honest. Security against a malicious Alice can be shown in an analogous way.

Let us take a minimal-OT \(G\) within some redundancy-free 2-party-function \(F\), such that for every minimal-OT \(\hat{G}\) in \(F(\Omega_A, \Omega_B(G))\) and every minimal-

---

\(^9\)We explicitely do not extend the \(\leadsto\)-relation to situations \((x, y)_F\) and \((x', y)_F\) with \(x \neq x'\), since this would destroy its transitivity.
OT $\tilde{G}$ in $F(\Omega_A(G), \Omega_B)$ holds\textsuperscript{10}:

$$V_A(\tilde{G}) \subseteq V_A(G) \Rightarrow V_A(\tilde{G}) = V_A(G)$$

$$V_B(\tilde{G}) \subseteq V_B(G) \Rightarrow V_B(\tilde{G}) = V_B(G)$$

By an appropriate consistent renaming of $F$ (cf. Definition 1) we can achieve $\Omega_A(G) = \Omega_B(G) = \{0, 1\}$ and:

$$G_A, G_B \in \left\{ \left( \begin{array}{cc} 0 & 0 \\ 0 & 0 \end{array} \right), \left( \begin{array}{cc} 0 & 0 \\ 0 & 1 \end{array} \right) \right\} \quad \text{(cf. Definition 4)}$$

Further w.l.o.g. we assume, that after this renaming the set $V_B(G)$ is the disjoint union of the following four sets:

$$\tilde{V}_0 := \left\{ y \in \tilde{V} \mid F_A(\{0, 1\}, \{y\}) = \left( \begin{array}{c} 0 \\ 0 \end{array} \right) \wedge F_B(\{0, 1\}, \{y\}) = \left( \begin{array}{c} 0 \\ 0 \end{array} \right) \right\}$$

$$\tilde{V}_1 := \left\{ y \in \tilde{V} \mid F_A(\{0, 1\}, \{y\}) = \left( \begin{array}{c} 0 \\ 0 \end{array} \right) \wedge F_B(\{0, 1\}, \{y\}) = \left( \begin{array}{c} 0 \\ 1 \end{array} \right) \right\}$$

$$\tilde{V}_2 := \left\{ y \in \tilde{V} \mid F_A(\{0, 1\}, \{y\}) = \left( \begin{array}{c} 0 \\ 1 \end{array} \right) \wedge F_B(\{0, 1\}, \{y\}) = \left( \begin{array}{c} 0 \\ 0 \end{array} \right) \right\}$$

$$\tilde{V}_3 := \left\{ y \in \tilde{V} \mid F_A(\{0, 1\}, \{y\}) = \left( \begin{array}{c} 0 \\ 1 \end{array} \right) \wedge F_B(\{0, 1\}, \{y\}) = \left( \begin{array}{c} 0 \\ 1 \end{array} \right) \right\}$$

For brevity we set $\tilde{V}' := \tilde{V}_1 \cup \tilde{V}_2 \cup \tilde{V}_3$. With this construction we can show that a malicious Bob can not deviate significantly from the input distribution specified by the protocol. We do this in six steps:

1. For every $x \in \Omega_A, y \in V_B(G)$ at least one of the following two assertions holds:

$$\forall y_0 \in \tilde{V}_0 : (x, y)_F \sim (x, y_0)_F$$

$$\forall y' \in \tilde{V}' : (x, y)_F \sim (x, y')_F$$

Otherwise we had some $x \in \Omega_A, y \in V_A(G), y_0 \in \tilde{V}_0$ and $y' \in \tilde{V}'$ with:

$$(x, y)_F \not\sim (x, y_0)_F \land (x, y)_F \not\sim (x, y')_F$$

\textsuperscript{10}It can be shown easily, that there always exists such a minimal-OT $G$ in $F$, as long as $F$ contains a minimal-OT at all: We just start with an arbitrary minimal-OT $G$ in $F$. If the required conditions are not met, we can take $\tilde{G}$ or $\tilde{G}$ as a new candidate and so on. This procedure does terminate, since in every step $V_A(G)$ or $V_B(G)$ is reduced at least by one element, while the other one remains exactly the same.
Hence in direct contradiction to our choice of $G$ we had a minimal-OT
$\hat{G} := F(\Omega_A(G), \{y_0, y_0\})$ with $y \in V_A(\hat{G}) \setminus V_A(G)$ and $V_A(\hat{G}) \subseteq V_A(G)$.

2. For every $y_0 \in \hat{V}_0 \setminus \{0\}$ there exists an $x \in \Omega_A$ with:

$$(x, y_0)_F \not\rightarrow (x, 0)_F \quad \land \forall y' \in V' : (x, y')_F \not\rightarrow (x, 0)_F$$

This follows from step 1, $F$ being redundancy-free and the transitivity of the "$\rightarrow$"-relation. Since $F$ is redundancy-free, we find some $x \in \Omega_A$ with $(x, y_0)_F \not\rightarrow (x, 0)_F$. Hence by step 1 follows:

$$\forall y' \in \hat{V}' : (x, y_0)_F \rightarrow (x, y')_F$$

Now if we could find some $y' \in \hat{V}'$ with $(x, y')_F \rightarrow (x, 0)_F$, in contradiction to our choice of $x$ this would imply $(x, y_0)_F \rightarrow (x, 0)_F$ because of the transitivity of the "$\rightarrow$"-Relation.

3. If Bob wants to use the input symbol “0” significantly less frequently than specified by the protocol, then for every $y_0 \in \hat{V}_0 \setminus \{0\}$ he has to use symbols from $\hat{V}_0 \setminus \{0, y_0\}$ instead. In step 2 we have seen that he can not use symbols from $\hat{V}' \cup \{y_0\}$.

4. Analogously to step 2 for every $y' \in \hat{V}' \setminus \{1\}$ there exists an $x \in \Omega_A$ with:

$$(x, y')_F \not\rightarrow (x, 1)_F \quad \land \forall y_0 \in V_0 : (x, y_0)_F \not\rightarrow (x, 1)_F$$

So if for some $y' \in \hat{V}' \setminus \{1\}$ Bob wants to use input symbols from $\hat{V}_0 \cup \{y'\}$ significantly more often than allowed by the protocol, then he has to reduce his input frequency of other input symbols than “1”. Since $n_B^{(k)}(\Omega_B \setminus \Omega_B(G))$ runs to zero with increasing security parameter $k$, this forces him to reduce his input frequency of the symbol “0”.

5. By combination of the previous two steps we can conclude: If for some $y' \in \hat{V}' \setminus \{1\}$ Bob wants to significantly increase his overall input frequency of symbols from $(\hat{V}_0 \cup \{y'\}) \setminus \{0\}$, then for every $y_0 \in \hat{V}_0 \setminus \{0\}$ he has to increase his overall input frequency of symbols from $\hat{V}_0 \setminus \{0, y_0\}$ roughly by the same amount. Thus neither symbols $y' \in \hat{V}' \setminus \{1\}$ nor symbols $y_0 \in \hat{V}_0 \setminus \{0\}$ can be used significantly too often by a malicious Bob without Alice catching him cheating.
6. In step 5 we have seen that a malicious Bob can not use any input symbol from $V_B(G) \setminus \Omega_B(G)$ significantly more often than specified by the protocol. The same holds for input symbols from $U_B(G)$. Furthermore, since $F$ is redundancy-free, there is no possibility that a malicious Bob might input “1” instead of “0” or vice versa too often without being caught cheating. Hence the considered offline-protocol is sufficiently secure.

So far for this informal overview. Section 3.1.1, Section 3.1.2 and Section 3.1.3 contain a formal proof of what we sketched here.

3.1.1 Stochastic basics

We start our proof by some very basic Definitions and Lemmata. This subsection somehow provides the raw material, we can build our proof of.

First of all we want to introduce some convenient but a bit sloppy use of the standard notion of negligibility.

Definition 12 (Negligibility).

1. A function $\mu : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is called negligible, if for every $n \in \mathbb{N}$ there is a $k_n \in \mathbb{N}$ with:

   $$\forall k \in \mathbb{N} : \ k > k_n \ \Rightarrow \ \mu(k) < k^n$$

2. Let an offline-protocol for a 2-party-function $F$ be given. We consider the following experiment: For a given security parameter $k$ a random protocol run $\lambda(k)$ is generated by a probabilistic algorithm that simulates some specified behaviour of Alice and Bob. A set $\Lambda$ of protocol runs itself is called negligible, if the function $\mu(k) := \Pr[\lambda(k) \in \Lambda]$ is negligible\(^{11}\).

Notation. Let a set $\Lambda$ of protocol runs and an assertion $A : \Lambda \to \{true, false\}$ be given. Further let $\tilde{\Lambda}$ be the set of all protocol runs $\lambda \in \Lambda$ with $A(\lambda) = false$. The assertion $A$ is said to hold for almost every protocol run $\lambda \in \Lambda$, if $\tilde{\Lambda}$ is negligible.

\(^{11}\)Hence whether a given set of protocol runs is negligible or not, may be highly dependent on the behaviour of Alice and Bob.
Our main tool for estimating statistic variables is Hoeffding’s Inequality. Since it seems not to be too well-known, it is posted here in the special form we need.

**Lemma 13** (Hoeffding’s Inequality). Let an integer \( k \in \mathbb{N} \) and a binomial or hypergeometrical distributed random variable \( X \) with expected value \( E(X) \) be given, such that \( \Pr[0 \leq X \leq k] = 1 \) holds. Then for every \( c \in \mathbb{R}_{>0} \) holds:

\[
\Pr[|X - E(X)| \geq c] \leq 2 \cdot \exp\left(\frac{-2c^2}{k}\right)
\]

Especially for every constant \( \Delta > \frac{1}{2} \) and every sequence \( (X_k)_{k \in \mathbb{N}} \) of binomial or hypergeometrical distributed random variables with \( \Pr[0 \leq X_k \leq k] = 1 \) the probability \( \Pr[|X_k - E(X_k)| \geq k\Delta] \) is negligible in \( k \).

**Proof.** The assertion of Lemma 13 is a direct consequence of Theorem 2 in [16] and the considerations in Chapter 6 of [16]. \( \square \)

Formally correct use of Lemma 13 also requires a detailed description of possible protocol runs by appropriate random variables. Within the following definition all needed random variables are listed.

**Notation.** A protocol run of an offline-protocol for a given 2-party function \( F \) is called complete, if it terminates regularly with the protocol step output.

For a given offline protocol \( \pi \) and some specified behaviour of the participating parties Alice and Bob we denote the set of all complete protocol runs by \( \Lambda(\pi) \). By \( k^\lambda \) we denote the security parameter of a given protocol run \( \lambda \). Furthermore for the protocol parameters we set \( n^\lambda_A := n^{(k^\lambda)}_A, n^\lambda_B := n^{(k^\lambda)}_B, c^\lambda := c^{(k^\lambda)} \) and \( \delta^\lambda := \delta^{(k^\lambda)} \) for convenience.

**Definition 14.** For a 2-party-function \( F \) let an offline-protocol \( \pi \) with parameters \( n_A, n_B, c \) and \( \delta \) be given. Further let some complete protocol run \( \lambda \) of this protocol \( \pi \) be given. By \( K^\lambda_1 := \{1, \ldots, k^\lambda\} \) we denote the state of \( K_A \)—that equals the state of \( K_B \)—just before protocol step Control A. By \( K^\lambda_2 := \{1, \ldots, k^\lambda\} \setminus K^\lambda_1 \) we denote the state of \( K_A \) between the protocol steps Control A and Control B and by \( K^\lambda_3 := \{1, \ldots, k^\lambda\} \setminus (K^\lambda_1 \cup K^\lambda_2) \) we denote the state of \( K_A \) right after protocol step Control B. By \( s^\lambda_A \) and \( s^\lambda_B \) we mean the state of this variables at the end of the protocol run. W.l.o.g. we assume, that the variables \( K_A, K_B, s^\lambda_A \) und \( s^\lambda_B \) are logged honestly even by
corrupted parties\textsuperscript{12}. For every \( x, x' \in \Omega_A, y, y' \in \Omega_B \) we define the following characteristics:

\( \hat{n}_1^\ell(x, y) \): the relative frequency of recorded situations \( (x, y)_F \) just before protocol step \textbf{Control A}:

\[
\hat{n}_1^\ell(x, y) := \frac{|s^\text{in}_A[K_1^\ell] \times s^\text{in}_B[K_1^\ell]|_{(x, y)}}{k^\ell}
\]

\( \hat{n}_2^\ell(x, y) \): the relative frequency of recorded situations \( (x, y)_F \) not tested so far between the protocol steps \textbf{Control A} and \textbf{Control B}:

\[
\hat{n}_2^\ell(x, y) := \frac{|s^\text{in}_A[K_2^\ell] \times s^\text{in}_B[K_2^\ell]|_{(x, y)}}{k^\ell - c^\ell}
\]

\( \hat{n}_3^\ell(x, y) \): the relative frequency of recorded situations \( (x, y)_F \) not tested at all after protocol step \textbf{Control B}:

\[
\hat{n}_3^\ell(x, y) := \frac{|s^\text{in}_A[K_3^\ell] \times s^\text{in}_B[K_3^\ell]|_{(x, y)}}{k^\ell - 2c^\ell}
\]

\( \hat{n}^\ell(x, y) \): the expected value of \( \hat{n}_1^\ell(x, y), \hat{n}_2^\ell(x, y) \) and \( \hat{n}_3^\ell(x, y) \), as long as both parties are honest:

\[
\hat{n}^\ell(x, y) := n_A^\ell(x) \cdot n_B^\ell(y)
\]

\( r_A^\ell(x, y) \): how often Alice asked Bob for his in- and output in a recorded situation \( (x, y)_F \) during \textbf{Control A}:

\[
r_A^\ell(x, y) := |s^\text{in}_A[K_A'] \times s^\text{in}_B[K_A']|_{(x, y)}
\]

\( l_B^\ell(x, y \leadsto y') \): how often Bob in a recorded situation \( (x, y)_F \) claimed \( y' \) to be his input:

\[
l_B^\ell(x, y \leadsto y') := |s^\text{in}_A[K_A'] \times s^\text{in}_B[K_A'] \times s^\text{in}_B[K_A']|_{(x, y, y')}
\]

\textsuperscript{12}This especially does not restrict the possibility of deviations from the specified input distributions in protocol step \textbf{Invocation of } \( F \), lies in protocol step \textbf{Control A} or \textbf{Control B}, dishonest choice of \( K_A' \) or \( K_B' \) and/or protocol abortions.
\( r_\lambda^B(x,y) \): how often Bob asked Alice for her in- and output in a recorded situation \((x,y)_F\) during Control B:

\[
r_\lambda^B(x,y) := |s_A^{\text{in}}[K'_B] \times s_B^{\text{in}}[K'_B]|_{(x,y)}
\]

\( l_\lambda^A(x \leadsto x', y) \): how often Alice in a recorded situation \((x,y)_F\) claimed \(x'\) to be her input:

\[
l_\lambda^A(x \leadsto x', y) := |s_A^{\text{in}}[K'_B] \times \hat{s}_A^{\text{in}}[K'_B] \times s_B^{\text{in}}[K'_B]|_{(x',x,y)}
\]

\( m^\lambda(x,y) \): the relative deviation of the protocol output from its expected value for honest parties:

\[
m^\lambda(x,y) := \hat{n}_3^\lambda(x,y) - \hat{n}^\lambda(x,y)
\]

As a conclusion of this subsection we sum up the most basic applications of Hoeffding’s Inequality (Lemma 13) to complete runs of offline-protocols.

**Corollary 15.** For a 2-party-function \( F \) let an offline-protocol \( \pi \) with parameters \( n_A, n_B, c \) and \( \delta \) be given as well as some \( \Delta > \frac{1}{2} \).

1. If Alice is honest, then for every \( S \subseteq \Omega_A, T \subseteq \Omega_B \) and almost every protocol run \( \lambda \in \Lambda(\pi) \) holds:

\[
(k^\lambda)^\Delta \geq |k^\lambda \cdot \hat{n}_1^\lambda(S, \Omega_B) - k^\lambda \cdot \hat{n}_A^\lambda(S)|
\]

\[
(k^\lambda)^\Delta \geq |k^\lambda \cdot \hat{n}_1^\lambda(S, T) - k^\lambda \cdot \hat{n}_1^\lambda(S, \Omega_B) \cdot \hat{n}_A^\lambda(\Omega_A, T)|
\]

\[
(k^\lambda)^\Delta \geq |r_A^\lambda(S, T) - c^\lambda \cdot \hat{n}_A^\lambda(S, T)|
\]

\[
(k^\lambda)^\Delta \geq |(k^\lambda - c^\lambda) \cdot \hat{n}_A^\lambda(S, T) - (k^\lambda - c^\lambda) \cdot \hat{n}_1^\lambda(S, T)|
\]

2. If Bob is honest, then for every \( S \subseteq \Omega_A, T \subseteq \Omega_B \) and almost every protocol run \( \lambda \in \Lambda(\pi) \) holds:

\[
(k^\lambda)^\Delta \geq |k^\lambda \cdot \hat{n}_1^\lambda(\Omega_A, T) - k^\lambda \cdot \hat{n}_B^\lambda(T)|
\]

\[
(k^\lambda)^\Delta \geq |k^\lambda \cdot \hat{n}_1^\lambda(S, T) - k^\lambda \cdot \hat{n}_1^\lambda(S, \Omega_B) \cdot \hat{n}_A^\lambda(\Omega_A, T)|
\]

\[
(k^\lambda)^\Delta \geq |r_B^\lambda(S, T) - c^\lambda \cdot \hat{n}_B^\lambda(S, T)|
\]

\[
(k^\lambda)^\Delta \geq |(k^\lambda - 2c^\lambda) \cdot \hat{n}_B^\lambda(S, T) - (k^\lambda - 2c^\lambda) \cdot \hat{n}_2^\lambda(S, T)|
\]

**Proof.** The assertions of Corollary 15 are direct consequences of Hoeffding’s Inequality (Lemma 13). \(\square\)
3.1.2 General properties of attacks on offline-protocols

In this subsection we make some more detailed assertions about attacks on offline-protocols. We need them for our main lemma (Lemma 23) in Section 3.1.3.

We start with a lower bound for the number of times a corrupted Bob has to lie in protocol step **Control A**.

**Lemma 16.** Let a 2-party-function $F$, some $\tilde{S}, S, S' \subseteq \Omega_A$, $\tilde{T}, T, \bar{T} \subseteq \Omega_B$ with $\bar{T} = \Omega_B \setminus T$, some $\varepsilon > 0$ and an offline-protocol $\pi \in \Pi_{\varepsilon}(\tilde{S}, \tilde{T})$ be given, such that for every security parameter $k$ holds:

$$n_A^\varepsilon(S) \leq n_A^\varepsilon(S') \neq 0$$

Further let Alice be honest. Then there exists a constant $C \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$l_B^\lambda(S, T \leadsto \bar{T}) \geq c^\lambda \cdot \left( \frac{n_A^\varepsilon(S) \cdot \hat{n}_A^\lambda(S', T)}{n_A^\varepsilon(S')} - 3c^\lambda - \hat{n}_A^\lambda(S, T) \right) - C \cdot (k^\lambda)^{1+\varepsilon}$$

**Proof.** We set $C := |S \times T| + \frac{7}{2}$. Let us consider some arbitrary protocol run $\lambda \in \Lambda(\pi)$ that fulfills the inequalities of Corollary 15.1. Since $\lambda$ is not aborted during protocol step **Control A**, we must have:

$$l_B^\lambda(S, \Omega_B \leadsto T) \leq c^\lambda \cdot \hat{n}_A^\lambda(S, T) + |S \times T| \cdot \delta^\lambda$$

Further by definition holds:

$$l_B^\lambda(S, T \leadsto \bar{T}) = r_A^\lambda(S, T) - l_B^\lambda(S, \Omega_B \leadsto T) + l_B^\lambda(S, \bar{T} \leadsto \bar{T})$$

Altogether this yields:

$$l_B^\lambda(S, T \leadsto \bar{T}) \geq r_A^\lambda(S, T) - \left( c^\lambda \cdot \hat{n}_A^\lambda(S, T) + |S \times T| \cdot (k^\lambda)^{1+\varepsilon} \right)$$

Hence we have to show\(^{13}\):

$$r_A^\lambda(S, T) > c^\lambda \cdot \left( \frac{n_A^\varepsilon(S) \cdot \hat{n}_A^\lambda(S', T)}{n_A^\varepsilon(S')} - 3c^\lambda \right) - \frac{7}{2} \cdot (k^\lambda)^{1+\varepsilon}$$

\(^{13}\)We even show a slightly stricter inequation but release it a bit to achieve symmetry between the assertions of Lemma 16 and Lemma 17.
We do this applying the inequalities from Corollary 15.1:

\[
    r^\lambda_A(S, T) \geq c^\lambda \cdot n^\lambda_A(S) \cdot \hat{n}^\lambda_T(O_A, T) - \left(1 + \frac{c^\lambda \cdot (1 + \hat{n}^\lambda_T(O_A, T))}{k^\lambda}\right) (k^\lambda)^{1+\varepsilon}
\]

\[
    > c^\lambda \cdot \left(\frac{n^\lambda_A(S) \cdot \hat{n}^\lambda_T(S', T)}{n^\lambda_A(S')} - \frac{2c^\lambda}{k^\lambda}\right) - \left(2 + \frac{3c^\lambda \cdot n^\lambda_A(S)}{k^\lambda \cdot n^\lambda_A(S')}(k^\lambda)^{1+\varepsilon}\right)
\]

In order to achieve symmetric assertions in Alice and Bob, we have to show an analogon to Lemma 16 for a possibly corrupted Alice.

**Lemma 17.** Let a 2-party-function \( F \), some \( \tilde{S}, S, \bar{S} \subseteq \Omega_A \), \( \tilde{T}, T, T' \subseteq \Omega_B \) with \( \bar{S} = \Omega_B \setminus S \), some \( \varepsilon > 0 \) and an offline-protocol \( \pi \in \Pi_F(\tilde{S}, \tilde{T}) \) be given, such that for every security parameter \( k \) holds:

\[
    n^{(k)}_B(T) \leq n^{(k)}_B(T') \neq 0
\]

Further let Bob be honest. Then there exists a constant \( C \in \mathbb{N} \), such that for almost every protocol run \( \lambda \in \Lambda(\pi) \) holds:

\[
    l^\lambda_B(S \mapsto \bar{S}, T) \geq r^\lambda_B(S, T) - (c^\lambda \cdot \hat{n}^\lambda(S, T) + |S \times T| \cdot \delta^\lambda)
\]

Hence we have to show:

\[
    r^\lambda_B(S, T) > c^\lambda \cdot \left(\frac{\hat{n}^\lambda_T(S, T') \cdot n^\lambda_B(T)}{n^\lambda_B(T')} - \frac{3c^\lambda}{k^\lambda}ight) - 6(k^\lambda)^{1+\varepsilon}
\]

We do this applying the inequalities from Corollary 15.2:

\[
    r^\lambda_B(S, T) > c^\lambda \cdot \left(\frac{k^\lambda \cdot \hat{n}^\lambda_1(S, \Omega_B) \cdot n^\lambda_B(T)}{k^\lambda - c^\lambda} - \frac{c^\lambda}{k^\lambda - c^\lambda}\right) - 3(k^\lambda)^{1+\varepsilon}
\]

\[
    > c^\lambda \cdot \left(\frac{k^\lambda \cdot \hat{n}^\lambda_3(S, T') \cdot n^\lambda_B(T)}{(k^\lambda - c^\lambda) \cdot n^\lambda_B(T')} - \frac{3c^\lambda}{k^\lambda - c^\lambda}\right) - 6(k^\lambda)^{1+\varepsilon}
\]

20
Since \( r_B^\lambda(S,T) \) cannot be negative, this yields:

\[
 r_B^\lambda(S,T) > c^\lambda \cdot \left( \frac{n_3^\lambda(S,T') \cdot n_B^\lambda(T)}{n_B^\lambda(T')} - \frac{3c^\lambda}{k^\lambda} - 6(k^\lambda)^{1+\varepsilon} \right)
\]

For symmetric reasons from here until the end of Section 3.1 w.l.o.g. we always may assume at least Alice to be honest. By the following corollary we extract from Lemma 16 what we need it for only.

**Corollary 18.** Let a 2-party-function \( F \), some \( \tilde{S}, \tilde{T}, T \subseteq \Omega_A \), \( \bar{T} := \Omega_B \setminus T \), \( x \in \Omega_A \), some \( \varepsilon > 0 \) and an offline-protocol \( \pi \in \Pi^\varepsilon_{\bar{F}}(\tilde{S}, \tilde{T}) \) be given. Further let Alice be honest. Then there exists a constant \( C \in \mathbb{N} \), such that for almost every protocol run \( \lambda \in \Lambda(\pi) \) holds:

\[
 l_B^\lambda(x,T \mapsto \bar{T}) > (k^\lambda)^{\frac{5+3\varepsilon}{8}} \cdot m^\lambda(\tilde{S}, T) - C \cdot (k^\lambda)^{\frac{1+\varepsilon}{2}}
\]

\[
 m^\lambda(S, \Omega_B) > -C \cdot (k^\lambda)^{\frac{1+\varepsilon}{2}}
\]

**Proof.** Both inequations of Corollary 18 can be derived from Definition 9 and Lemma 16 quite easily. With \( \tilde{C} \in \mathbb{N} \) great enough they follow from:

\[
 l_B^\lambda(x,T \mapsto \bar{T}) > c^\lambda \cdot \left( \frac{n_3^\lambda(x) \cdot \hat{n}_3^\lambda(\tilde{S}, T)}{n_A^\lambda(\tilde{S})} - \frac{3c^\lambda}{k^\lambda} - \hat{n}^\lambda(x, T) \right) - \tilde{C} \cdot (k^\lambda)^{\frac{1+\varepsilon}{2}}
\]

\[
 0 > c^\lambda \cdot \left( \hat{n}_3^\lambda(\Omega_A \setminus S, \Omega_A) - \frac{3c^\lambda}{k^\lambda} - \hat{n}^\lambda(\Omega_A \setminus S, \Omega_A) \right) - \tilde{C} \cdot (k^\lambda)^{\frac{1+\varepsilon}{2}}
\]

We already know, how often a corrupted Bob has to lie, if he wants to manipulate the protocol’s output characteristics in a specific way. We still have to find limitations for his possibilities of lying without being caught by Alice.

**Lemma 19.** Let a 2-party-function \( F \), some \( x \in \Omega_A \), some \( y,y' \in \Omega_B \) with \( (x,y)_F \not\sim (x,y')_F \), some \( \varepsilon > 0 \) and an offline-protocol \( \pi \) be given. Further let Alice be honest. Then for almost every protocol run \( \lambda \in \Lambda(\pi) \) holds:

\[
 l_B^\lambda(x, y \mapsto y') < \frac{(k^\lambda)^\varepsilon}{\min\{n_\lambda^\lambda(\tilde{x}) \mid \tilde{x} \in \Omega_A\}}
\]

**Proof.** For our proof let us consider some arbitrary protocol run \( \lambda \in \Lambda(\pi) \) with \( l_B^\lambda(x, y \mapsto y') > 0 \), i.e. we have a malicious Bob, that for several times in
a situation \((x, y)_F\) claims his input to be \(y'\) and his output to be \(F_B(x, y')\). Because of the assumption \((x, y)_F \not\Rightarrow (x, y')_F\), there also exists an \(x' \in \Omega_B\) with \(F_B(x', y) = F_B(x, y)\) and \((F_A(x', y), F_B(x, y')) \neq (F_A(x', y'), F_B(x', y'))\). Thus Bob is caught cheating immediately, if he behaves that way in a situation \((x', y)_F\), too. Hence Bob has to differentiate between situations \((x, y)_F\) and \((x', y)_F\) just by guessing and we can estimate his overall probability of success from above by:

\[
(1 - n^λ_A(x'))^l^B(x,y\rightarrow y')
\]

This implies the assertion of Lemma 19, what directly follows from:

\[
1 > \exp(-1) \geq \limsup_{k \to \infty} \left(1 - n^λ_A(x')\right)^{\left(n^λ_A(x')\right)^{-1}}
\]

Now we can present a first general limitation of the output characteristics of offline-protocols.

**Corollary 20.** Let a 2-party-function \(F\) be given as well as some \(\tilde{S} \subseteq \Omega_A\), \(\tilde{x} \in \Omega_A\) and \(T, T \subseteq \Omega_B\), such that for \(\bar{T} := \Omega_B \setminus T\) holds:

\[
\forall y \in T, y' \in \bar{T} : (\tilde{x}, y)_F \not\Rightarrow (\tilde{x}, y')_F.
\]

Further let some \(\varepsilon > 0\) and an offline-protocol \(\pi \in \Pi_ε(\tilde{S}, \bar{T})\) be given. Let Alice be honest. Then there exists a constant \(C \in \mathbb{N}\), such that for almost every protocol run \(\lambda \in \Lambda(\pi)\) holds:

\[
m^λ(\tilde{S}, T) < C \cdot (k^λ)^{-\frac{1-\varepsilon}{8}}
\]

**Proof.** We just have to combine Corollary 18, Definition 9 and a direct implication of Lemma 19, whereby we find a constant \(\hat{C} \in \mathbb{N}\), such that for almost every protocol run \(\lambda \in \Lambda(\pi)\) holds:

\[
\begin{align*}
P_B^A(\tilde{x}, T \sim \bar{T}) &> (k^λ)^{\frac{3+4\varepsilon}{8}} \cdot m^λ(\tilde{S}, T) - \hat{C} \cdot (k^λ)^{\frac{1+\varepsilon}{2}} \\
(k^λ)^{-\frac{1-\varepsilon}{8}} &\leq \min\{n^λ_A(x) \mid x \in \Omega_A\} \\
P_B^A(\tilde{x}, T \sim \bar{T}) &< \frac{(k^λ)^{\frac{3+4\varepsilon}{8}}}{\min\{n^λ_A(x) \mid x \in \Omega_A\}}
\end{align*}
\]

This directly gives us the assertion of Corollary 20. \(\square\)
For our main lemma (Lemma 23) in Section 3.1.3 we especially need the following guarantee about possible malicious input distributions concerning the sets of Definition 11.

**Lemma 21.** Let a 2-party-function $F$, some $\tilde{S} \subseteq \Omega_A$, $\tilde{T} \subseteq \Omega_B$, some $\varepsilon > 0$ and an offline-protocol $\pi \in \Pi_\varepsilon^F(\tilde{S}, \tilde{T})$ be given with. Further let Alice be honest. Then there exists a constant $C \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$m^\lambda(\tilde{S}, U_B(\tilde{T})) < C \cdot (k^\lambda)^{-\frac{1-\varepsilon}{8}}$$

**Proof.** We construct a covering $U_B(\tilde{T}) = \bigcup_{x \in \Omega_A} \hat{U}_x$ by:

$$\forall x \in \Omega_A : \quad \hat{U}_x := \{ y \in \Omega_B : \forall y' \in \tilde{T} : (x,y)_F \not\rightarrow (x,y')_F \}$$

Further for every protocol run $\lambda \in \Lambda(\pi)$ let us take some $\tilde{x}(\lambda) \in \Omega_A$, such that holds:

$$\hat{n}_3^\lambda(\tilde{S}, \hat{U}_{\tilde{x}(\lambda)}) = \max\{ \hat{n}_3^\lambda(\tilde{S}, \hat{U}_x) \mid x \in \Omega_A \}$$

Hence we have for every protocol run $\lambda \in \Lambda(\pi)$:

$$m^\lambda(\tilde{S}, U_B(\tilde{T})) \leq \sum_{x \in \Omega_A} \hat{n}_3^\lambda(\tilde{S}, \hat{U}_x) \leq |\Omega_A| \cdot \left( m^\lambda(\tilde{S}, \hat{U}_{\tilde{x}(\lambda)}) + n_B^\lambda(\hat{U}_{\tilde{x}(\lambda)}) \right)$$

Now on the one hand from $\hat{U}_{\tilde{x}(\lambda)} \cap \tilde{T} = \emptyset$ by Definition 9 for every protocol run $\lambda \in \Lambda(\pi)$ follows:

$$n_B^\lambda(\hat{U}_{\tilde{x}(\lambda)}) = |\hat{U}_{\tilde{x}(\lambda)}| \cdot (k^\lambda)^{-\frac{1-\varepsilon}{8}} \leq |\Omega_B| \cdot (k^\lambda)^{-\frac{1-\varepsilon}{8}}$$

On the other hand by the transitivity of the "$\not\rightarrow$"-relation follows:

$$\forall x \in \Omega_A, y \in \hat{U}_x, y' \in \Omega_B \setminus \hat{U}_x : (x,y)_F \not\rightarrow (x,y')_F$$

Hence by Corollary 20 we find a constant $\hat{C} \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$m^\lambda(\tilde{S}, \hat{U}_{\tilde{x}(\lambda)}) < \hat{C} \cdot (k^\lambda)^{-\frac{1-\varepsilon}{8}}$$

Altogether this yields for almost every protocol run $\lambda \in \Lambda(\pi)$:

$$m^\lambda(\tilde{S}, U_B(\tilde{T})) < |\Omega_A| \cdot (\hat{C} + |\Omega_B|) \cdot (k^\lambda)^{-\frac{1-\varepsilon}{8}} \quad \square$$
3.1.3 Resistant offline-protocols

In this subsection we formally define, what we mean by sufficient secure offline-protocols. Then we take the pieces from the previous two subsections and put them together to a security proof.

**Definition 22** (Resistance). Let a 2-party-function $F$ be given. A minimal-OT $G$ in $F$ is called resistant, if for every $\varepsilon > 0$ and every offline-protocol $\pi \in \Pi_F(\Omega_A(G),\Omega_B(G))$ there exists a resistance-constant $C \in \mathbb{N}$ with the following properties:

1. If Alice follows the protocol honestly, then for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:
   $$\forall y \in \Omega_B(G) : \quad m^\lambda(\Omega_A(G), y) > -C \cdot (k^\lambda)^{-\frac{1-\varepsilon}{2}}$$

2. If Bob follows the protocol honestly, then for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:
   $$\forall x \in \Omega_A(G) : \quad m^\lambda(x, \Omega_B(G)) > -C \cdot (k^\lambda)^{-\frac{1-\varepsilon}{2}}$$

3. If both parties follow the protocol honestly, then for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:
   $$\forall S \subseteq \Omega_A, T \subseteq \Omega_B : \quad |m^\lambda(S,T)| < (k^\lambda)^{-\frac{1-\varepsilon}{2}}$$

In this context an offline-protocol $\pi \in \Pi_F(\Omega_A(G),\Omega_B(G))$ itself is called resistant, if $G$ is a resistant minimal-OT.

**Lemma 23.** Let a redundancy-free 2-party-function $F$ be given, that contains a minimal-OT $\tilde{G}$. Let Alice be honest. Then there exists a minimal-OT $G$ in $F$ with $\Omega_A(G) = \Omega_A(\tilde{G})$, such that for every $\tilde{S} \subseteq \Omega_A$, every $\varepsilon > 0$ and every offline-protocol $\pi \in \Pi_F(\tilde{S},\Omega_B(G))$ there exists a constant $C \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$\forall y \in \Omega_B(G) : \quad m^\lambda(\tilde{S}, y) > -C \cdot (k^\lambda)^{-\frac{1-\varepsilon}{2}}$$

**Proof.** First of all we define $\Gamma$ to be the set of all minimal-OTs $G'$ in $F$ with $\Omega_A(G') = \Omega_A(\tilde{G})$. Let us pick some fixed minimal-OT $G \in \Gamma$ with:

$$\forall G' \in \Gamma : \quad V_B(G') \subseteq V_B(G) \Rightarrow V_B(G') = V_B(G)$$

24
Now let some $\tilde{S} \subseteq \Omega_A$, $\varepsilon > 0$ and an offline-protocol $\pi \in \Pi_F(\tilde{S}, \Omega_B(G))$ be given. We have to find a constant $C \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$\forall y \in \Omega_B(G): \quad m^\lambda(\tilde{S}, y) > -C \cdot (k^\lambda)^{-\frac{\varepsilon}{8}}$$

By an appropriate consistent renaming of $F$ we achieve (cf. Definition 4):

$$\Omega_A(G) = \Omega_B(G) = \{0, 1\} \land F_B(0, 0) = F_B(1, 0)$$

Now we divide $V_B(G)$ into two subsets $\tilde{V}_0$ and $\tilde{V}' := V_B(G) \setminus \tilde{V}_0$ with:

$$\tilde{V}_0 := \{ y \in V_B(G) \mid F_B(0, y) = F_B(1, y) \land \forall x \in \Omega_A(G) : F_A(x, y) = F_A(x, 0) \}$$

Moreover for every $\tilde{x} \in \Omega_A$ and $\tilde{y} \in \Omega_B$ we define:

$$\hat{U}_{\tilde{x}}(\tilde{y}) := \{ y \in \Omega_B \mid (\tilde{x}, y)_F \not\sim (\tilde{x}, \tilde{y})_F \}$$

$$\hat{V}_{\tilde{x}}(\tilde{y}) := \{ y \in \Omega_B \mid (\tilde{x}, y)_F \sim (\tilde{x}, \tilde{y})_F \}$$

We will need these sets throughout this proof for several times. Let us now start with the main proof, which is split up into six steps, each one consisting of a small claim followed by its own small proof.

1. First of all we want to prove, that for every tuple of input symbols $(x, y) \in \Omega_A \times V_B(G)$ at least one of the following two assertions holds:

$$\forall y_0 \in \tilde{V}_0 : \quad (x, y)_F \sim (x, y_0)_F$$

$$\forall y' \in \tilde{V}' : \quad (x, y)_F \not\sim (x, y')_F$$

Proof. We prove this by contradiction. Thus let us assume, we could find some $x \in \Omega_A$, $y \in V_B(G)$, $y_0 \in \tilde{V}_0$ and $y' \in \tilde{V}'$ with:

$$(x, y)_F \not\leftrightarrow (x, y_0)_F \land (x, y)_F \not\leftrightarrow (x, y')_F$$

By this assumption we get a minimal-OT $\hat{G} := F(\Omega_A(G), \{y_0, y'\})$ with $y \in V_B(G) \setminus V_B(\hat{G})$. Furthermore we can derive $V_B(\hat{G}) \subseteq V_B(G)$ from $\Omega_B(\hat{G}) \subseteq V_B(G)$ and the transitivity of the “$\sim$”-relation. Altogether this yields a direct contradiction to our choice of $G$. 

25
2. For every $y_0 \in \tilde{V}_0 \setminus \{0\}$ there exists an $x \in \Omega_A$ with:

$$(x, y_0)_F \not\rightarrow (x, 0)_F \land \forall y' \in V': (x, y')_F \not\rightarrow (x, 0)_F$$

Proof. Let some $y_0 \in \tilde{V}_0 \setminus \{0\}$ be given. Since $F$ is redundancy-free, we find some $x \in \Omega_A$ with $(x, y_0)_F \not\rightarrow (x, 0)_F$. Hence by step 1 follows:

$$\forall y' \in \tilde{V}' : (x, y_0)_F \not\rightarrow (x, y')_F$$

Now if we could find some $y' \in \tilde{V}'$ with $(x, y')_F \not\rightarrow (x, 0)_F$, in contradiction to our choice of $x$ this would imply $(x, y_0)_F \not\rightarrow (x, 0)_F$ because of the transitivity of the "\not\rightarrow"-Relation.

3. There exists a constant $C \in \mathbb{N}$, such that for every $y_0 \in \tilde{V}_0 \setminus \{0\}$ and almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$m^\lambda(\tilde{S}, 0) > -\hat{n}_3(\tilde{S}, \tilde{V}_0 \setminus \{0, y_0\}) - C \cdot (k^\lambda)^{\frac{1-\varepsilon}{8}}$$

Proof. It suffices to find a constant $\hat{C}_y$ for each $y \in \tilde{V}_0 \setminus \{0\}$, since with those constants we can set $C := \max\{\hat{C}_y \mid y \in \tilde{V}_0 \setminus \{0\}\}$. So let us pick some arbitrary $y_0 \in \tilde{V}_0 \setminus \{0\}$. By step 2 we find some $x \in \Omega_A$ with:

$$\tilde{V}' \cup \{y_0\} \subseteq \hat{U}_x(0)$$

That is:

$$\hat{V}_x(0) = \Omega_B \setminus \hat{U}_x(0) \subseteq \Omega_B \setminus (\tilde{V}' \cup \{y_0\}) = U_B(G) \cup \tilde{V}_0 \setminus \{y_0\}$$

By Definition 14 for every protocol run $\lambda \in \Lambda(\pi)$ follows:

$$m^\lambda(\tilde{S}, 0) = m^\lambda(\tilde{S}, \Omega_B) - m^\lambda(\tilde{S}, \hat{U}_x(0)) - m^\lambda(\tilde{S}, \hat{V}_x(0) \setminus \{0\})$$

$$\geq m^\lambda(\tilde{S}, \Omega_B) - m^\lambda(\tilde{S}, \hat{U}_x(0)) - \hat{n}_3(\tilde{S}, \hat{V}_x(0) \setminus \{0\})$$

$$\geq m^\lambda(\tilde{S}, \Omega_B) - m^\lambda(\tilde{S}, \hat{U}_x(0)) - \hat{n}_3(\tilde{S}, U_B(G) \cup \tilde{V}_0 \setminus \{0, y_0\})$$

By Corollary 18, Corollary 20 and Lemma 21 together with Definition 9 we can estimate this as claimed.
4. There exists a constant $C \in \mathbb{N}$, such that for every $y' \in \tilde{V} \setminus \{1\}$ and almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$m^\lambda(\tilde{S}, 0) < -\hat{n}_3^\lambda(\tilde{S}, \tilde{V}_0 \setminus \{0\}) - \hat{n}_3^\lambda(\tilde{S}, y') + C \cdot (k^\lambda)^{-\frac{1-\varepsilon}{8}}$$

**Proof.** Again it suffices to find a constant $\hat{C}_y$ for each $y \in \tilde{V} \setminus \{1\}$. So let us pick some arbitrary $y' \in \tilde{V}' \setminus \{1\}$. Analogously to step 2 we find some $x \in \Omega_A$ with:

$$\tilde{V}_0 \cup \{y'\} \subseteq \hat{U}_x(1)$$

Thus by Definition 14 and Definition 9 for every complete protocol run $\lambda \in \Lambda(\pi)$ holds:

$$m^\lambda(\tilde{S}, \hat{U}_x(1) \setminus \{0\}) > \hat{n}_3^\lambda(\tilde{S}, (\tilde{V}_0 \cup \{y'\}) \setminus \{0\}) - |\Omega_B| \cdot (k^\lambda)^{-\frac{1-\varepsilon}{8}}$$

Besides by Corollary 20 we find some constant $\tilde{C} \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$m^\lambda(\tilde{S}, \hat{U}_x(1)) < \tilde{C} \cdot (k^\lambda)^{-\frac{1-\varepsilon}{8}}$$

Altogether this yields for almost every protocol run $\lambda \in \Lambda(\pi)$:

$$m^\lambda(\tilde{S}, 0) < -\hat{n}_3^\lambda(\tilde{S}, (\tilde{V}_0 \cup \{y'\}) \setminus \{0\}) + (\hat{C} + |\Omega_B|) \cdot (k^\lambda)^{-\frac{1-\varepsilon}{8}}$$

5. There exists a constant $C \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$m^\lambda(\tilde{S}, \Omega_B(G)) > C \cdot (k^\lambda)^{-\frac{1-\varepsilon}{8}}$$

**Proof.** By combining step 3 and step 4 we find some constant $\tilde{C} \in \mathbb{N}$, such that for every $y_0 \in \tilde{V}_0 \setminus \{0\}$, every $y' \in \tilde{V}' \setminus \{1\}$ and almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$\hat{n}_3^\lambda(\tilde{S}, \{y_0, y'\}) < \tilde{C} \cdot (k^\lambda)^{-\frac{1-\varepsilon}{8}}$$

Thus we have for almost every protocol run $\lambda \in \Lambda(\pi)$:

$$m^\lambda(\tilde{S}, V_B(G) \setminus \Omega_B(G)) \leq \hat{n}_3^\lambda(\tilde{S}, V_B(G) \setminus \Omega_B(G)) < |\Omega_B| \cdot \tilde{C} \cdot (k^\lambda)^{-\frac{1-\varepsilon}{8}}$$
By Corollary 18 and Lemma 21 we find some constants $\hat{C}, \bar{C} \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ follows:

$$m^\lambda(\tilde{S}, \Omega_B(G)) = m^\lambda(\tilde{S}, \Omega_B) - m^\lambda(\tilde{S}, U_B(G)) - m^\lambda(\tilde{S}, V_B(G) \backslash \Omega_B(G)) \geq - (\hat{C} + \bar{C} + |\Omega_B| \cdot \bar{C}) \cdot (k^\lambda)^{-\frac{1-\epsilon}{8}}$$

6. There exists a constant $C \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$\forall y \in \Omega_B(G) : m^\lambda(\tilde{S}, y) > -C \cdot (k^\lambda)^{-\frac{1-\epsilon}{8}}$$

Proof. Since $F$ is redundancy-free, we find some $x \in \Omega_A$ with:

$$(x, 0)_F \not\rightarrow (x, 1)_F \text{ i.e. } \hat{U}_x(1) \cap \Omega_B(G) = \{0\}$$

Thus by Definition 14 and Definition 9 for every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$m^\lambda(\tilde{S}, \hat{U}_x(1) \backslash \{0\}) \geq -n^\lambda(\tilde{S}, \hat{U}_x(1) \backslash \{0\}) = -(\hat{U}_x(1) - 1) \cdot (k^\lambda)^{-\frac{1-\epsilon}{8}}$$

Hence by step 5 and Corollary 20 we find some constants $\hat{C}, \bar{C} \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$m^\lambda(\tilde{S}, 1) = m^\lambda(\tilde{S}, \Omega_B(G)) - m^\lambda(\tilde{S}, \hat{U}_x(1)) + m^\lambda(\tilde{S}, \hat{U}_x(1) \backslash \{0\}) \geq -(\hat{C} + \bar{C} + |\Omega_B|) \cdot (k^\lambda)^{-\frac{1-\epsilon}{8}}$$

Analogously we can find some constants $\hat{C}', \bar{C}' \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ holds:

$$m^\lambda(\tilde{S}, 0) > -(\hat{C}' + \bar{C}' + |\Omega_B|) \cdot (k^\lambda)^{-\frac{1-\epsilon}{8}}$$

Corollary 24. Every redundancy-free 2-party-function containing a minimal-OT also does contain a resistant minimal-OT.
Proof. Let a redundancy-free 2-party-function $F$ be given that contains a minimal-OT $\tilde{G}$. By Lemma 23 we find a minimal-OT $\hat{G}$ in $F$, such that for every $\tilde{S} \subseteq \Omega_A$, every $\varepsilon > 0$ and every offline-protocol $\pi \in \Pi_\varepsilon(\tilde{S}, \Omega_B(\hat{G}))$ there exists a constant $\hat{C}_1 \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ with honest behaviour of Alice holds:

$$\forall y \in \Omega_B(\hat{G}) : \; m^\lambda(\tilde{S}, y) > -\hat{C}_1 \cdot (k^\lambda)^{-\frac{1+\varepsilon}{8}}$$

In a second step analogously to Lemma 23 we find a minimal-OT $G$ in $F$ with $\Omega_B(G) = \Omega_B(\hat{G})$, such that for every $\varepsilon > 0$ and every offline-protocol $\pi \in \Pi_\varepsilon(\Omega_A(G), \Omega_B(G))$ there exists a constant $\hat{C}_2 \in \mathbb{N}$, such that for almost every protocol run $\lambda \in \Lambda(\pi)$ with honest behaviour of Bob holds:

$$\forall x \in \Omega_A(G) : \; m^\lambda(x, \Omega_B(G)) > -\hat{C}_2 \cdot (k^\lambda)^{-\frac{1+\varepsilon}{8}}$$

Finally by Definition 8, Definition 9 and Hoeffding’s Inequality (Lemma 13) follows, that for every $\varepsilon > 0$, every offline-protocol $\pi \in \Pi_\varepsilon(\Omega_A(G), \Omega_B(G))$ and almost every protocol run $\lambda \in \Lambda(\pi)$ with honest behaviour of both parties holds:

$$\forall S \subseteq \Omega_A, T \subseteq \Omega_B : \; \left| m^\lambda(S, T) \right| < (k^\lambda)^{-\frac{1+\varepsilon}{2}}$$

Hence $G$ is a resistant minimal-OT.

\[ \square \]

3.2 Reduction of OT to resistant offline-protocols

Section 3.1 was about secure generation of correlated data. Now we want to build an invocation of OT from such data. For this purpose we present a reduction protocol that mainly is inspired by the protocol used in [8] for reduction of OT to noisy channels. We adopted several parts and the main structure of that protocol.

3.2.1 The reduction protocol

Here we present a protocol that implements an invocation of $(\frac{\varepsilon}{2})$-OT by use of a resistant offline-protocol. Let $b_0, b_1 \in \{0, 1\}$ be Alice’s OT-input and let $c \in \{0, 1\}$ be Bob’s OT-input. Further let some $\varepsilon \in (0, 1)$ and some $\Delta \in (\frac{\varepsilon}{8}, 1)$ be given as well as a 2-party-function $F$ that contains
a resistant minimal-OT $G$ with resistance-constant $C$. W.l.o.g. we assume $\Omega_A(G) = \Omega_B(G) = \{0, 1\}$ and:

$$G_A, G_B \in \left\{ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \right\} \quad (\text{cf. Definition 4})$$

These three possibilities of $G$ each require its own protocol variant:

**Case 1:** $G_A = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$, $G_B = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$

Let an offline-protocol $\pi \in \Pi_F^\varepsilon(\{0, 1\}, \{0, 1\})$ be given whose input distributions fulfill:

$$n_A^{(k)}(0) = n_A^{(k)}(1) = \frac{1}{2} \left( 1 - (|\Omega_A| - 2) \cdot k^{-\frac{1 - \varepsilon}{2}} \right)$$

$$n_B^{(k)}(0) = n_B^{(k)}(1) = \frac{1}{2} \left( 1 - (|\Omega_B| - 2) \cdot k^{-\frac{1 - \varepsilon}{2}} \right)$$

1. Alice chooses two random bit strings $\tilde{b}_0, \tilde{b}_1 \in \{0, 1\}^{\lceil k\Delta \rceil}$ with:

$$\bigoplus_{i=1}^{\lceil k\Delta \rceil} \tilde{b}_0[i] = b_0 \quad \text{and} \quad \forall i \in \{1, \ldots, \lceil k\Delta \rceil\} : \tilde{b}_0[i] \oplus \tilde{b}_1[i] = b_0 \oplus b_1$$

Bob chooses a random bit string $\tilde{c} \in \{0, 1\}^{\lceil k\Delta \rceil}$ with:

$$\bigoplus_{i=1}^{\lceil k\Delta \rceil} \tilde{c}[i] = c$$

2. Alice and Bob execute the offline-protocol $\pi$ with security parameter $k$. Let $s_A$ and $s_B$ be their output strings.

3. Positions that correspond to input symbols from $\Omega_A \setminus \Omega_A(G)$ or $\Omega_B \setminus \Omega_B(G)$ are revealed and deleted from $s_A$ and $s_B$. If more than $(1 - \tilde{n}^{(k)}(\{0, 1\}, \{0, 1\})) \cdot k + k^{\frac{1 - \varepsilon}{2}}$ positions are deleted in total, the protocol is aborted. Furthermore Alice aborts the protocol, if there is left a position $i$ with:

$$s_A[i] \not\in \{(0, 0), (1, 0)\}$$

Bob aborts the protocol, if there is left a position $i$ with:

$$s_A[i] \not\in \{(0, 0), (1, 0), (1, 1)\}$$

30
4. Alice randomly permutes her string $s_A$. She sends a description of this permutation to Bob, who permutates his string $s_B$ in the same way.

5. From $s_A$ and $s_B$ Alice and Bob locally generate $\lceil k^\Delta \rceil$ strings $\tilde{s}_A^{(1)}, \ldots, \tilde{s}_A^{(\lceil k^\Delta \rceil)}$ and $\tilde{s}_B^{(1)}, \ldots, \tilde{s}_B^{(\lceil k^\Delta \rceil)}$ of equal length $l := \left\lceil \frac{|s_A|}{k^\Delta} \right\rceil$, such that for every $i \in \{1, \ldots, l\}$ and every $j \in \{1, \ldots, \lceil k^\Delta \rceil\}$ holds:

$$s_A \left[ i + (j-1) \cdot l \right] = (0, 0) \iff \tilde{s}_A^{(j)}[i] = 0$$
$$s_A \left[ i + (j-1) \cdot l \right] = (1, 0) \iff \tilde{s}_A^{(j)}[i] = 1$$
$$s_B \left[ i + (j-1) \cdot l \right] = (0, 0) \iff \tilde{s}_B^{(j)}[i] = \bot$$
$$s_B \left[ i + (j-1) \cdot l \right] = (1, 0) \iff \tilde{s}_B^{(j)}[i] = 0$$
$$s_B \left[ i + (j-1) \cdot l \right] = (1, 1) \iff \tilde{s}_B^{(j)}[i] = 1$$

6. For every $j \in \{1, \ldots, \lceil k^\Delta \rceil\}$ the following subprotocol is executed:
   (a) Bob chooses some random sets $K_0^{(j)}, K_1^{(j)} \subseteq \{1, \ldots, l\}$ with:

   $$|K_0^{(j)}| = |K_1^{(j)}| = \left\lceil \frac{3l}{8} \right\rceil$$
   $$K_0^{(j)} \cap K_1^{(j)} = \emptyset$$
   $$\forall i \in K_0^{(j)} : \tilde{s}_B^{(j)}[i] \neq \bot$$

   He announces them to Alice, who aborts the protocol, if one of the first two conditions is violated.
   (b) Alice chooses a random bit string $\tilde{b}^{(j)} \in \{0, 1\}^l$ with:

   $$\bigoplus_{i \in K_0^{(j)}} \tilde{b}^{(j)}[i] = \tilde{b}_0[j] \quad \text{and} \quad \bigoplus_{i \in K_1^{(j)}} \tilde{b}^{(j)}[i] = \tilde{b}_1[j]$$

   Then she generates a bit string $\hat{b}^{(j)} \in \{0, 1\}^l$ by:

   $$\forall i \in \{1, \ldots, l\} : \hat{b}^{(j)}[i] := \tilde{b}^{(j)}[i] \oplus \tilde{s}_A^{(j)}[i]$$

   (c) She sends $\hat{b}^{(j)}$ to Bob, who reconstructs:

   $$\tilde{b}_{\hat{j}}[j] = \bigoplus_{i \in K_0^{(j)}} \tilde{s}_B^{(j)}[i] \oplus \hat{b}^{(j)}[i]$$
7. Bob computes his output $b_c$ by:

$$b_c = \bigoplus_{j=1}^{\lceil k\Delta \rceil} \tilde{b}_{c[j]}[j]$$

**Case 2:** $G_A = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$, $G_B = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$

Since we do not bother about the direction of the implemented OT, we can reduce this case to the one above.

**Case 3:** $G_A = G_B = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$

Let an offline-protocol $\pi \in \Pi_F^e(\{0,1\},\{0,1\})$ be given whose input distributions fulfill:

$$n_A^{(k)}(0) = \frac{1}{3} \left( 1 - (|\Omega_A| - 2) \cdot k^{-\frac{1+\epsilon}{3}} \right)$$
$$n_A^{(k)}(1) = \frac{2}{3} \left( 1 - (|\Omega_A| - 2) \cdot k^{-\frac{1+\epsilon}{3}} \right)$$
$$n_B^{(k)}(0) = n_B^{(k)}(1) = \frac{1}{2} \left( 1 - (|\Omega_B| - 2) \cdot k^{-\frac{1+\epsilon}{3}} \right)$$

1. Alice chooses two random bit strings $\tilde{b}_0, \tilde{b}_1 \in \{0,1\}^{\lceil k\Delta \rceil}$ with:

$$\bigoplus_{i=1}^{\lceil k\Delta \rceil} \tilde{b}_0[i] = b_0 \quad \text{and} \quad \forall i \in \{1, \ldots, \lceil k\Delta \rceil\} : \tilde{b}_0[i] \oplus \tilde{b}_1[i] = b_0 \oplus b_1$$

Bob chooses a random bit string $\tilde{c} \in \{0,1\}^{\lceil k\Delta \rceil}$ with:

$$\bigoplus_{i=1}^{\lceil k\Delta \rceil} \tilde{c}[i] = c$$

2. Alice and Bob execute the offline-protocol $\pi$ with security parameter $k$. Let $s_A$ and $s_B$ be their output strings.

3. Positions that correspond to input symbols from $\Omega_A \setminus \Omega_A(G)$ or $\Omega_B \setminus \Omega_B(G)$ are revealed and deleted from $s_A$ and $s_B$. If more than $(1 - \tilde{n}^{(k)}(\{0,1\},\{0,1\})) \cdot k + k^{\frac{1+\epsilon}{3}}$ positions are deleted in total,
the protocol is aborted. Furthermore Alice aborts the protocol, if there is left a position $i$ with:

$$s_A[i] \not\in \{(0,0), (1,0), (1,1)\}$$

Bob aborts the protocol, if there is left a position $i$ with:

$$s_A[i] \not\in \{(0,0), (1,0), (1,1)\}$$

4. (a) Alice and Bob locally rename and delete entries of $s_A$ and $s_B$ by the following scheme:
   - Every entry $(0,0)$ is renamed to $0$.
   - Every entry $(1,0)$ is renamed to $1$.
   - Every entry $(1,1)$ is deleted.

(b) Alice deletes $\lvert s_A \rvert_0 - \lvert s_A \rvert_1$ positions from $s_A$, such that after this deletion holds:

$$\lvert s_A \rvert_0 = \lvert s_A \rvert_1$$

She announces these positions to Bob, who deletes them from $s_B$, too. Bob aborts the protocol, if he has to delete more than $2k_{\frac{1+\epsilon}{2}}$ positions in this protocol step.

(c) Alice randomly permutates the entries of $s_A$, such that for every $i \in \{1, \ldots, \frac{\lvert s_A \rvert}{2}\}$ after this permutation holds:

$$s_A[2i-1] \neq s_A[2i]$$

She announces the permutation to Bob, who permutates $s_B$ the same way. Bob aborts the protocol, if after this permutation he finds some $i \in \{1, \ldots, \frac{\lvert s_B \rvert}{2}\}$ with:

$$s_B[2i-1] = s_B[2i] = 1$$

5. From $s_A$ and $s_B$ Alice and Bob locally generate $\lceil k^\Delta \rceil$ strings $\tilde{s}_A^{(1)}, \ldots, \tilde{s}_A^{(\lceil k^\Delta \rceil)}$ and $\tilde{s}_B^{(1)}, \ldots, \tilde{s}_B^{(\lceil k^\Delta \rceil)}$ of equal length $l := \left\lfloor \frac{\lvert s_A \rvert}{2k^\Delta} \right\rfloor$, such that for every $i \in \{1, \ldots, l\}$, every $j \in \{1, \ldots, \lceil k^\Delta \rceil\}$ and
\[ h_{i,j} := 2(i + (j - 1) \cdot l) \] holds:

\[
\begin{align*}
(s_A[h_{i,j} - 1], s_A[h_{i,j}]) &= (0, 1) \iff \tilde{s}_A^{(j)}[i] = 0 \\
(s_A[h_{i,j} - 1], s_A[h_{i,j}]) &= (1, 0) \iff \tilde{s}_A^{(j)}[i] = 1 \\
(s_B[h_{i,j} - 1], s_B[h_{i,j}]) &= (0, 0) \iff \tilde{s}_B^{(j)}[i] = \perp \\
(s_B[h_{i,j} - 1], s_B[h_{i,j}]) &= (1, 0) \iff \tilde{s}_B^{(j)}[i] = 0 \\
(s_B[h_{i,j} - 1], s_B[h_{i,j}]) &= (0, 1) \iff \tilde{s}_B^{(j)}[i] = 1
\end{align*}
\]

6. For every \( j \in \{1, \ldots, \lceil k^\Delta \rceil \} \) the following subprotocol is executed:

(a) Bob chooses some random sets \( K_0^{(j)}, K_1^{(j)} \subseteq \{1, \ldots, l\} \) with:

\[
|K_0^{(j)}| = |K_1^{(j)}| = \left\lceil \frac{3l}{8} \right\rceil \\
K_0^{(j)} \cap K_1^{(j)} = \emptyset \\
\forall i \in K_0^{(j)} \cup K_1^{(j)} : \tilde{s}_B^{(j)}[i] \neq \perp
\]

He announces them to Alice, who aborts the protocol, if one of the first two conditions is violated.

(b) Alice chooses a random bit string \( \bar{b}^{(j)} \in \{0, 1\}^l \) with:

\[
\bigoplus_{i \in K_0^{(j)}} \bar{b}^{(j)}[i] = \bar{b}_0[j] \quad \text{and} \quad \bigoplus_{i \in K_1^{(j)}} \bar{b}^{(j)}[i] = \bar{b}_1[j]
\]

Then she generates a bit string \( \hat{b}^{(j)} \in \{0, 1\}^l \) by:

\[
\forall i \in \{1, \ldots, l\} : \quad \hat{b}^{(j)}[i] := \bar{b}^{(j)}[i] \oplus \tilde{s}_A^{(j)}[i]
\]

(c) She sends \( \hat{b}^{(j)} \) to Bob, who reconstructs:

\[
\tilde{b}^{(j)}[j] = \bigoplus_{i \in K_0^{(j)}} \tilde{s}_B^{(j)}[i] \oplus \hat{b}^{(j)}[i]
\]

7. Bob computes his output \( b_c \) by:

\[
b_c = \bigoplus_{j=1}^{\lceil k^\Delta \rceil} \tilde{b}^{(j)}[j]
\]
3.2.2 Correctness of the reduction

It is not hard to verify, that the protocol from Section 3.2.1 does what it is supposed to do, as long as no party is corrupted. From Hoeffding’s Inequality (Lemma 13), Definition 22.3 and our choice of the protocol parameters in Section 3.2.1 quite simply follows, that the protocol in a totally uncorrupted case is aborted only with negligible probability. It is straight through to see, too, that an uncorrupted Bob does learn \( b_c \) at the end of each non-aborted protocol run with an uncorrupted Alice.

3.2.3 Security against a malicious Alice

In order to prove security against a malicious Alice we have to construct a Simulator \( S \) that simulates the real model for an Alice totally controlled by the environment \( Z \) and that gives the right values \( b_0 \) and \( b_1 \) to the ideal functionality \( \mathcal{F} \).

For this purpose \( S \) takes a random bit \( c \) for Bob’s OT-input and begins to simulate internally a real protocol run, while Alice is controlled by the environment. If the simulated Bob aborts the protocol, the simulator terminates. Else he learns \( b_c \) in protocol step 7. By the following procedure he then tries to compute \( b_{c-1} \):

1. From the in- and output of the simulated 2-party-function \( F \) our simulator \( S \) for every \( j \in \{1, \ldots, \lceil k^\Delta \rceil \} \) generates a string \( \tilde{s}_A^{(j)} \) like an honest Alice would have done. Wherever this is impossible due to dishonest behaviour of Alice, he uses a special symbol “*”.

2. He chooses some \( j' \in \{1, \ldots, \lceil k^\Delta \rceil \} \) with:

   \[
   \forall i \in K_0^{(j')} \cup K_1^{(j')} : \quad \tilde{s}_A^{(j')}[i] \neq *
   \]

   If this is impossible, \( S \) terminates.

3. He computes:

   \[
   b_{1-c} = b_c \oplus \bigoplus_{i \in K_0^{(j')} \cup K_1^{(j')}} s_A^{(j')}[i]
   \]

Now we have to see, why this simulator makes the ideal model indistinguishable from the real model.
Proof. The simulated Bob aborts the protocol exactly when a real Bob also would do so. Hence w.l.o.g. we can assume, that the simulated protocol run is aborted only with non-overwhelming probability. Moreover we may restrict our considerations to simulations with non-aborted protocol runs.

First of all we have to show, that the simulator $S$ only with negligible probability fails in his search for $j'$. If he does, this especially means:

$$m_* := \sum_{j=1}^{[k^\Delta]} |\tilde{s}_A^{(j)}|_* \geq [k^\Delta]$$

Depending on $G$ beeing symmetric or not we have to distinguish between two cases:

1. If $G$ is asymmetric, then $m_*$ is limited by the number of inputs from $\Omega_A \setminus \Omega_A(G)$ in protocol step 2. Hence by Definition 22.2 we find some constant $C$, such that with overwhelming probability holds:

$$m_* < C \cdot k^{\frac{7+\varepsilon}{8}}$$

For $k$ great enough this is a contradiction to $m_* \geq [k^\Delta]$.

2. If $G$ is symmetric, we have to consider protocol step 4 aswell. Let us imagine, that just before protocol step 4.(a) from his protocol transcript the simulator generates the string $s_A$ like an uncorrupted Alice would have done. Then by Hoeffding’s Inequality (Lemma 13) and Definition 22.2 we find some constant $\hat{C}$, such that with overwhelming probability holds:

$$\frac{k}{3} - \hat{C} \cdot k^{\frac{7+\varepsilon}{8}} \leq |s_A|_{(0,0)} \leq \frac{k}{3} + \hat{C} \cdot k^{\frac{7+\varepsilon}{8}}$$
$$\frac{k}{3} - \hat{C} \cdot k^{\frac{7+\varepsilon}{8}} \leq |s_A|_{(1,0)} \leq \frac{k}{3} + \hat{C} \cdot k^{\frac{7+\varepsilon}{8}}$$
$$\frac{k}{3} - \hat{C} \cdot k^{\frac{7+\varepsilon}{8}} \leq |s_A|_{(1,1)} \leq \frac{k}{3} + \hat{C} \cdot k^{\frac{7+\varepsilon}{8}}$$

Now in step 4.(a) the simulator can rename and delete entries of $s_A$ by the following scheme:

- Every entry $(0,0)$ is renamed to 0.
- Every entry $(1,0)$ is renamed to 1.
• Every entry \((1, 1)\) is deleted.
• Every other entry is renamed to \(*\).

Further let \(\tilde{m}_*\) be the number of tuples \((s_A[2i-1], s_A[2i]) = (0, 0)\) after step 4.(c). This number \(\tilde{m}_*\) must be subpolynomial in \(k\), since Alice is caught cheating in step 4.(c) with some probability of at least:

\[
1 - \left(1 - \left(\frac{|s_A \times s_B|(0,1)}{|s_A|_0}\right)^2\right)^{\tilde{m}_*}
\]

Hence together with our considerations above we find some constant \(C\), such that after protocol step 4.(c) we can estimate:

\[
m_* \leq \frac{|s_A|}{2} - \left(|s_A|_0 - |s_A|_* - 2\tilde{m}_*\right) \leq C \cdot k^{\frac{7+\varepsilon}{8}}
\]

In total this also gives us a contradiction to \(m_* \geq \lceil k\Delta \rceil\) for \(k\) great enough.

Now if the environment \(Z\) could distinguish between real and ideal model better than with some negligible advantage, this would mean that after a non-aborted simulation \(Z\) would decide to be in the ideal model with some probability greater than 50\%. Since this is impossible whenever the simulator did guess right Bob’s input \(c\), the environment especially could distinguish whether \(S\) did guess right or wrong with some non-zero advantage. In contradiction to this, even with the help of an oracle, that knows \(\tilde{c}[j]\) for every \(j \in \{1, \ldots, \lceil k\Delta \rceil\}\) the environment cannot gather any information about \(\tilde{c}[j']\) during a non-aborted simulation. \(\square\)

3.2.4 Security against a malicious Bob

In order to prove security against a malicious Bob we have to construct a Simulator \(S\) that simulates the real model for a Bob totally controlled by the environment \(Z\) and that gives the right value \(c\) to the ideal functionality \(F\).

The simulator internally simulates a real protocol run with Alice’s inputs \(b_0\) and \(b_1\) initialized by random bits. Corrupted Bob is totally controled by \(Z\). If simulated Alice aborts the protocol, the simulator also terminates. Direktly before Alice generates the bit string \(\tilde{b}(\lceil k\Delta \rceil)\) in protocol step 6.(b), the simulator revises \(\tilde{b}_0[\lceil k\Delta \rceil]\) and \(\tilde{b}_1[\lceil k\Delta \rceil]\) by the following algorithm:
1. From the in- and output of the simulated 2-party-function $F$ our simulator $S$ for every $j \in \{1, \ldots, \lceil k^{\Delta} \rceil \}$ generates a string $\tilde{s}_B^{(j)}$ like an honest Bob would have done. Wherever this is impossible due to dishonest behaviour of Bob, he uses a special symbol “∗”.

2. The simulator choses a bit string $\tilde{c} \in \{0, 1\}^\lceil k^\Delta \rceil$ with:

$$\forall j \in \{1, \ldots, \lceil k^\Delta \rceil \}, \ i \in K_{\tilde{c}[j]} :\quad \tilde{s}_B^{(j)}[i] \neq \bot$$

If this is impossible, $S$ sets:

$$\forall j \in \{1, \ldots, \lceil k^\Delta \rceil \} : \quad \tilde{c}[j] := 0$$

3. The simulator computes:

$$c = \bigoplus_{j=1}^{\lceil k^\Delta \rceil} \tilde{c}[j]$$

4. The bit $c$ is sent to the ideal functionality and $S$ receives some bit $b'$.

5. Now $S$ can revise $\tilde{b}_0[\lceil k^\Delta \rceil]$ and $\tilde{b}_1[\lceil k^\Delta \rceil]$ by:

$$\tilde{b}_{\tilde{c}[\lceil k^\Delta \rceil]}[\lceil k^\Delta \rceil] := b' \oplus \bigoplus_{j=1}^{\lceil k^\Delta \rceil} \tilde{b}_{\tilde{c}[j]}[j]$$

$$\tilde{b}_{1-\tilde{c}[\lceil k^\Delta \rceil]}[\lceil k^\Delta \rceil] := \tilde{b}_{\tilde{c}[\lceil k^\Delta \rceil]}[\lceil k^\Delta \rceil] \oplus b_0 \oplus b_1$$

Again we have to show, that this simulator makes the ideal model indistinguishable from the real model.

**Proof.** If simulated Alice aborts the protocol, the environment $Z$ has no information whether this was a protocol run in the ideal model or in the real model, since such an abortion happens independently of Alice’s inputs $b_0$ and $b_1$. Hence w.l.o.g. we can assume again, that the simulated protocol run is aborted only with non-overwhelming probability. Moreover we may restrict our considerations to simulations with non-aborted protocol runs.

Obviously the environment $Z$ cannot distinguish between a protocol run in the real model and a protocol run in the ideal model where the simulator did guess right the XOR of Alice’s inputs. Hence if the environment could
distinguish between real and ideal model better than with some negligible advantage, this would mean that after a non-aborted simulation with non-negligible probability the environment has some information whether the simulator did guess right or wrong. Hence there would have to be some $j \in \{1, \ldots, \lceil k^\Delta \rceil \}$ with:

$$|\tilde{s}_B^{(j)}|_\perp < |\tilde{s}_B^{(j)}| - 2 \cdot \left[ \frac{3 \cdot |\tilde{s}_B^{(j)}|}{8} \right] \leq |\tilde{s}_B^{(j)}|_4$$

In contradiction to this by Hoeffding’s Inequality (Lemma 13) and Definition 22.1 one can derive quite easily, that for every constant $d > 0$ after a non-aborted simulation with overwhelming probability holds:

$$\forall j \in \{1, \ldots, \lceil k^\Delta \rceil \} : |\tilde{s}_B^{(j)}|_\perp > \frac{|\tilde{s}_B^{(j)}|}{2 + d} \quad \square$$

### 3.2.5 Passive adversaries

If there is only a passive adversary, then according to Hoeffding’s Inequality (Lemma 13) every offline-protocol is resistant in the sense of Definition 22, even if the underlying 2-party-function $F$ is not redundancy-free. Hence by the protocol from Section 3.2.1 OT can be reduced to an arbitrary 2-party-function that contains a minimal-OT.

### 4 Conclusion

We introduced a new class of deterministic and stateless 2-party-primitives with finite input alphabets, that not only contains symmetric and asymmetric 2-party-functions. For this class of cryptographic primitives we gave comprehensive completeness criteria with respect to active and passive security. These completeness criteria especially show that every 2-party-function can be transformed canonically into a symmetric one, if it is not complete. This also implies that every essentially non-symmetric 2-party-function is as hard to implement as OT itself. As far as we know, similar and/or related questions at least partly are still open for several other classes of cryptographic primitives:

**Non-deterministic primitives:** These primitives give output that is randomly distributed. Most measurements of physical processes do so.
Unfair primitives: Corrupted parties may use some extended input alphabets. This is motivated by potential side channel attacks.

Primitives with infinite input alphabets: The input alphabets may be continuous. Most parameters in physical experiments are continuous, too.

Multi-party-primitives: In real life there are a lot of protocols with more than just two participants.

Stateful primitives: The considered primitives might be more powerful like finite automata or even turing machines.

Completeness in synchronous security models: As far as we know this topic has been investigated much less intensively than completeness in asynchronous models.
References


