Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Robust Combiners for Cryptographic Primitives

Christian Sommer

Master's Thesis in Computer Science

April 1st, 2006 – September 30th, 2006

Supervisors: Prof. Dr. Ueli Maurer
Bartosz Przydatek

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science ETH in Computer Science *at ETH Zürich (Swiss Federal Institute of Technology Zurich).*

Acknowledgements

First of all, I would like to thank Bartosz Przydatek for giving me valuable insights into combiners and Ueli Maurer for passionately teaching Cryptography and Information Theory lectures.

I am grateful for the valuable discussions that I had with the brilliant members of the Cryptography and Information Security Research Group and the Quantum Information Research Group.

This thesis was improved substantially by the valuable comments of those who read preliminary versions of it. In particular, I would like to thank Viktor Galliard, Stephen Hogan, Bartosz Przydatek, Peter Schneider, Hansruedi and Hermine Sommer, Stefano Tessaro, and Jürg Wullschleger for their helpful suggestions and proofreading. (The remaining errors and omissions are entirely the author's responsibility.)

This work is dedicated to my family and friends, who have always given me great support during my studies at ETH (October 2002 – September 2006).

Zürich, September 30th, 2006.

Christian Sommer

Abstract

In cryptography, we do not know which computational assumptions are the most secure to rely on. Robust combiners attempt to solve this problem. Given several implementations of a certain primitive, e.g., of a commitment scheme, a combiner merges them into a new implementation that is secure if a minimum number of the input implementations are secure. A $(k; n)$-robust combiner merges n implementations, where k of them are required to remain secure. In this thesis, we investigate combiners for various primitives. We show which combiners for commitment schemes are possible and which combiners do not exist. We show that a certain combiner construction is impossible if only half of the input implementations are secure (technically speaking, we prove that *transparent black-box* $(1; 2)$-robust combiners for commitment schemes do not exist). Furthermore, we give explicit constructions for combiners where the majority of the input implementations are assumed to be secure.

We make further investigations about combiners for interactive proof systems. However, this scenario is far more complicated and therefore, the statements made are somewhat crude.

For oblivious transfer, a yet unpublished paper of Meier *et al.* proposes more tolerant constructions using a "swap" operation. We show that such an operation is necessary for certain types of combiners.

Zusammenfassung

Combiners werden in der Kryptographie verwendet, um sich gegen ungewisse berechenmässige Annahmen abzusichern. Gegeben sind Implementationen einer Primitive, zum Beispiel Bit Commitment. Ein Combiner verknüpft diese Implementationen so, dass die Kombination sicher ist, wenn mindestens eine gewisse Anzahl der gegebenen Implementationen sicher ist. In dieser Arbeit befassen wir uns mit Combinern für verschiedene Primitiven. Wir untersuchen, welche Combiners für Commitment Schemes existieren und welche unmöglich sind. Wir zeigen, dass *transparent black-box* Combiner Konstruktionen nur möglich sind, falls die Mehrheit der gegebenen Implementationen sicher ist und konstruieren einen solchen Combiner.

Weiter haben wir uns mit Combinern für interaktive Beweissysteme befasst. Diese Ausgangslage ist viel komplizierter und die gemachten Aussagen sind eher einfacher Natur.

Für Oblivious Transfer werden in einer noch unveröffentlichten Arbeit von Meier *et al.* tolerantere Konstruktionen vorgeschlagen, welche nur mit einer "swap"-Operation funktionieren. Wir zeigen, dass eine Operation dieser Art notwendig ist.

Contents

1 Introduction

An implementation of a cryptographic primitive is often based on some (unproven) computational assumption, e.g., the hardness of factoring integer numbers or the hardness of computing discrete logarithms. Unfortunately, both in the design and in the use of such primitives, we do not know which assumptions are the most secure to rely on. We would like to have an implementation of the primitive that is as secure as possible given the current state of knowledge.

As it is often unclear which of the assumptions is most likely to be correct, just picking a single implementation usually does not work - we might bet on the wrong assumption. A better option would therefore be to have an implementation that is guaranteed to be secure as long as a minimum number of the assumptions are correct. A $(k; n)$-robust combiner is given n implementations of which k can be assumed to be secure. By merging these input schemes in an intelligent manner, a secure implementation can be obtained.

1.1 Our contributions

Our contributions pertain to combiners and impossibility proofs of commitment schemes, interactive proofs, and oblivious transfer.

Commitment schemes

In Section 3, we show which combiners for commitment schemes are possible and which cannot exist. There exists a black-box combiner that is secure even if only one implementation is secure (Theorem 5). This reduction is already known [IL89, Nao91, HILL99, Her05]. However, for transparent black-box combiners, the majority of the input implementations need to be secure. The primary contribution of this thesis is a proof that there is no transparent black-box combiner if only half of the implementations are secure (Theorem 4 and Corollary 4). Fortunately, there are simple combiners if the binding or hiding property is guaranteed (Theorem 1 and 2), or if the majority of the input implementations are secure ([Her05] and Theorem 6).

Interactive proof systems

Proof systems are far more complicated. We state our approach and some basic results about combiners for interactive proof systems in Section 4. The results reveal that a possible transparent black-box combiner will not

be deterministic, and that any function that splits input x into multiple instances $x_1 \dots x_n$ must not rely on the witness w.

Oblivious transfer

Our contribution is a modification of [HKN+05, Theorem 4.1] to prove that uni-directional transparent black-box $\{3, 2\}$-robust uniform combiners do not exist (Theorem 14).

1.2 Related work

The idea of using several implementations for encryption in order to obtain better security has been a research topic in cryptography for several years [AB81, EG85, Sie85, DK05]. Recent efforts in research on combiners have also investigated interactive primitives. [Her05] analysed combiner constructions on an abstract level and introduced combiners for one-way functions and commitment schemes. Furthermore, Herzberg pointed out that efficiency is crucial for combiners. Combiners for oblivious transfer, key-agreement, and public-key encryption were studied in [HKN+05]. Combiners for private information retrieval and general "cross-primitive" combiners were presented in [MP06]. Security assumptions for combiners were adapted and combiner definitions were strengthened in [MPW06].

2 Preliminaries

2.1 Cryptographic primitives

Notation Cryptographic primitives are indicated in boldface capitals as in **T**. An arbitrary implementation of **T** (satisfying **T**'s functionality) is denoted by a typewriter font letter as in $\mathtt{t}$. For a cryptographic property a of a primitive **T**, $\mathtt{t}^a$ denotes an arbitrary implementation of **T** satisfying functionality and a. In this work, we usually do not distinguish information theoretic and cryptographic properties. A capital letter (A) is used if we explicitly consider an information theoretic property. For lowercase letters, the property might be cryptographic or even information theoretic secure. For all security properties a_i of **T**, $\mathbf{T} = \{\mathtt{t}^{a_1 a_2 \cdots}\}$.

For primitives we distinguish between functional properties and security properties. Roughly speaking, functional properties mean that an implementation fulfils the primitive's protocol specification. Functionality usually can be tested. Cryptography's main objective is the fulfilment of the security properties, which specify, e.g., under which assumptions a party can compute information about a value.

One-way functions (**OWF**) are the basic primitives in cryptography. Most other primitives can be related to one-way functions. Often one assumes the existence of one-way functions in order to perform cryptography.

Definition 1 (Strong One-Way Functions [Gol00])**.** *A function $f : \{0,1\}^* \to \{0,1\}^*$ is called* (strongly) one-way *if the following two conditions hold:*

i. Easy to compute: *There exists a (deterministic) polynomial time algorithm A such that on input x algorithm A outputs $f(x)$ (i.e., $A(x) = f(x)$).*

ii. Hard to invert: *U_n denotes a random variable uniformly distributed over $\{0,1\}^n$. For every probabilistic polynomial time algorithm A', every positive polynomial $p(\cdot)$, and all sufficiently large n,*

$$Pr[A'(f(U_n), 1^n) \in f^{-1}(f(U_n))] < \frac{1}{p(n)}$$

According to the notation above, $\mathtt{owf}^o$ denotes a function satisfying the *o*ne-way property (hard to invert). We consider efficient computability as a functional property of one-way functions.

Oblivious transfer (**OT**) [Rab81, EGL85] is used by various multi-party computations. There are several variants of oblivious transfer; the version we use within this thesis is called *chosen one-out-of-two oblivious transfer*.

Definition 2 (Oblivious Transfer [WW04])**.** *By* oblivious transfer *we mean the following primitive between a sender A and a receiver B. A has two inputs b_0 and b_1 and no output. B has input c and output b_c.*

An implementation of **OT** is denoted by $\mathtt{ot}$. We abbreviate an execution of an **OT** protocol by $\mathtt{ot}(b_0, b_1; c)$. The security properties are *S*ender's privacy and *R*eceiver's privacy.

Commitment schemes (**BC**) are an important tool in cryptographic protocols. Such a scheme works in two phases. In the commit phase, the sender (Alice) can commit to a value b without telling the receiver (Bob) her choice. The receiver usually gets a commitment c. Bob should not be able to guess b efficiently. This is called the *hiding* property. In a later open phase, Alice reveals her choice. Bob gets an output b' and decides whether he accepts the opening of the commitment. Alice should not be able to change her bit afterwards, i.e., to convince Bob of another bit $b' \neq b$. This is called the *binding* property. For an introduction to commitment schemes, see for example [Dam98]. The following definition is adapted from [Gol00].

Definition 3 (Bit Commitment Scheme)**.** *A* bit commitment scheme *is a pair of probabilistic polynomial time interactive machines, denoted (S, R) (for* sender *and* receiver*), satisfying the following:*

- Input specification: *The* common input *is an integer n presented in unary (serving as the security parameter).*
  The private input *to the sender is a bit, denoted b.*
- Secrecy (or hiding): *The receiver (even when deviating arbitrarily from the protocol) is unable to distinguish a commitment to* 0 *from a commitment to* 1*. Indistinguishability can be computational, statistical, or information theoretic.*
- Unambiguity (or binding): *Preliminaries to the requirement*
  - *i.* A receiver's view of an interaction *with the sender, denoted $(\rho_r, \bar{m})$, consists of the random coins used by the receiver (ρ_r) and the sequence of messages received from the sender ($\bar{m}$).*
  - *ii. Let $\sigma \in \{0, 1\}$. We say that a receiver's view (of such interaction), $(\rho_r, \bar{m})$ is a* possible σ-commitment *if there exists a string ρ_s such*

that $\bar{m}$ describes the messages received by R when R uses local coins ρ_r and interacts with machine S that uses local coins ρ_s and has input $(\sigma, 1^n)$.

iii. We say that the receiver's view $(\rho_r, \bar{m})$ is ambiguous *if it is both a possible* 0*-commitment and a possible* 1*-commitment.*

The unambiguity requirement *asserts that it is either computationally hard for the sender to find a sequence of messages that, together with these coin tosses, forms an ambiguous receiver view, or that it is statistically or information theoretic impossible to find such a sequence.*

According to the notation above, $\mathtt{bc}^{bh}$ denotes a commitment scheme satisfying both the *B*inding and *H*iding property.

It is well known that for any commitment to a bit, at most one of the properties, either binding or hiding, can be information theoretic secure at the same time. We give a brief proof.

Theorem. *No commitment is both information theoretic hiding and information theoretic binding.*

Proof. Let B and C be random variables for the bit and the commitment. H is the entropy function.

- Information theoretic hiding means that the commitment C contains no information about the bit B, i.e., $H(B|C) = H(B)$.
- Information theoretic binding means that the bit B is determined uniquely having the commitment C, i.e., $H(B|C) = 0$.

These two requirements lead to $H(B) = 0$. This is impossible for any meaningful commitment. □

Thus, the best possible implementations are those where one property is information theoretic / statistically secure and the other property is based on a computational assumption. Implementations of this kind are the most common.

Definition 4. *A* statistically binding bit commitment scheme *is a bit commitment scheme with statistical unambiguity and computational secrecy.*

Definition 5. *A* statistically hiding bit commitment scheme *is a bit commitment scheme with computational unambiguity and statistical secrecy.*

An unbounded sender can cheat in cryptographic binding schemes by breaking the security assumption. An unbounded sender can sometimes cheat a bounded receiver even in secure binding schemes. Consider the folklore reduction from **OT** to **BC**: $b = x_0 \oplus x_1$, COMMIT : $\mathtt{ot}(x_0, x_1; c)$, the receiver gets x_c and the sender does not know which bit the receiver got; OPEN : send x_0, x_1. We assume an **OT** with cryptographic privacy properties. The receiver can cheat if he obtains both bits. The sender can cheat if and only if he knows which bit the receiver chose and if the receiver is unable to cheat. Therefore, the security assumption against the sender depends on the capabilities of both the sender and the receiver. This commitment scheme does not conform to the previous definitions.

Loosely speaking, *zero-knowledge proofs* [GMR85, GMW91] are proofs that yield nothing beyond the validity of the assertion. Common mathematical proofs are static. *Interactive proofs* are more like a dynamic proof where a verifier, if he is convinced, accepts a sequence of arguments and answers to his interrogation of the prover.

Definition 6 (Interactive Proof System [Gol00])**.** *A pair of interactive Turing machines* $\mathtt{ip} = (P, V)$ *is called an* interactive proof system for a language L *if machine* V *is polynomial time and the following two conditions hold* ($\langle P, V\rangle(x)$ *denotes the random variable representing the (local) output of* V *when interacting with* P *on common input* x*, 1 is interpreted as "accept", 0 is interpreted as "reject"):*

- Completeness: *For every* $x \in L$*,*

$$Pr[\langle P, V\rangle(x) = 1] \geq \frac{2}{3}$$

- Soundness: *For every* $x \notin L$ *and every interactive machine* P'*,*

$$Pr[\langle P', V\rangle(x) = 1] \leq \frac{1}{3}$$

Definition 7 (Perfect Zero-Knowledge [Gol00])**.** *Let* (P, V) *be an interactive proof system for a language* L*. We say that* (P, V) *is* perfect zero-knowledge *if for every probabilistic polynomial time interactive machine* V^* *there exists a probabilistic polynomial time algorithm* M^* *such that for every* $x \in L$ *the following two conditions hold:*

i. With probability at most $\frac{1}{2}$*, on input* x*, machine* M^* *outputs a special symbol denoted* $\perp$ *(i.e.,* $Pr[M^*(x) = \perp] \leq \frac{1}{2}$*).*

ii. Let $m^*(x)$ be a random variable describing the distribution of $M^*(x)$ conditioned on $M^*(x) \neq \perp$ (i.e., $Pr[M^*(x) = \alpha] = Pr[m^*(x) = \alpha | M^*(x) \neq \perp]$ for every $\alpha \in \{0,1\}^*$). Then the following random variables are identically distributed:

- *$\langle P, V^* \rangle (x)$ (i.e., the output of the interactive machine V^* after interacting with the interactive machine P on common input x)*
- *$m^*(x)$ (i.e., the output of machine M^* on input x, conditioned on not being $\perp$)*

Machine M^* is called a perfect simulator *for the interaction of V^* with P.*

Definition 8 (Computational Zero-Knowledge [Gol00])**.** *Let (P, V) be an interactive proof system for a language L. We say that (P, V) is* computational zero-knowledge *(or just* zero-knowledge*) if for every probabilistic polynomial time interactive machine V^* there exists a probabilistic polynomial time algorithm M^* such that the following two ensembles are computationally indistinguishable:*

- *$\{\langle P, V^* \rangle (x)\}_{x \in L}$ (i.e., the output of the interactive machine V^* after it interacts with the interactive machine P on common input x)*
- *$\{M^*(x)\}_{x \in L}$ (i.e., the output of machine M^* on input x)*

Machine M^* is called a simulator *for the interaction of V^* with P.*

Definition 9 (Zero-Knowledge Interactive Proof System)**.** *A* zero-knowledge interactive proof system for a language L *is an interactive proof system for L that satisfies the zero-knowledge conditions.*

According to the notation above, $\mathtt{ip}^{csz}$ denotes a proof system satisfying *C*ompleteness, *S*oundness, and the *Z*ero-knowledge property. A superscript w stands for *W*itness-hiding.

2.2 Combiners

Combiners compound several input implementations of a certain primitive in order to obtain a secure implementation. Given some assumptions about the input implementations, security properties about the combination are proven. Thus, combiners form a tolerant reduction.

Notation A combiner for $\mathbf{T}$ is denoted with $\mathtt{t}_{cmb}(\ldots)$. The input configuration is taken arbitrarily from a set of tuples of implementations, $S = \{(\mathtt{t}^{\cdots}, \mathtt{t}^{\cdots}, \ldots), \ldots\}$. To denote the set of all permutations of a tuple $(\mathtt{t}^{\cdots}, \mathtt{t}^{\cdots}, \ldots)$ we use $\mathcal{P}(\mathtt{t}^{\cdots}, \mathtt{t}^{\cdots}, \ldots)$, e.g., $\mathcal{P}(\mathtt{t}^A, \mathtt{t}) = \{(\mathtt{t}^A, \mathtt{t}), (\mathtt{t}, \mathtt{t}^A)\}$.

Definition 10 ($(k;n)$-Robust Combiner [HKN+05])**.** *Let* $\mathbf{T}$ *be a cryptographic primitive. A* $(k;n)$-robust combiner *for* $\mathbf{T}$ *is a probabilistic polynomial time Turing machine that gets* n *candidate schemes*[1] *as inputs, and implements* $\mathbf{T}$ *while satisfying the following two properties:*

i. If at least k *candidates securely implement* $\mathbf{T}$ *then the combiner also securely implements* $\mathbf{T}$*.*

ii. The running time of the combiner is polynomial in the security parameter (m*), in* n*, and in the lengths of the inputs to* $\mathbf{T}$*.*

If there is a combiner from a set of tuples of input implementations S to a set of tuples of output implementations S' (usually, $|S'| = 1$), this is a reduction $S \Longrightarrow S'$. E.g., a $(1;2)$-robust combiner for $\mathbf{T}$ with property a ($\mathbf{T} = \{\mathtt{t}^a\}$) performs the reduction $\{(\mathtt{t}^a, \mathtt{t}), (\mathtt{t}, \mathtt{t}^a)\} \Longrightarrow \{\mathtt{t}^a\} = \mathbf{T}$.

Definition 11 (Black-Box Combiner [HKN+05])**.** *A* $(k;n)$*-robust combiner is said to be* black-box *if the following conditions hold:*

i. Black-box implementation*: The combiner is an oracle probabilistic polynomial time Turing machine given access to the candidates via oracle calls to their implementation function.*

ii. Black-box proof*: For at least* $n-k+1$ *candidates there exists an oracle probabilistic polynomial time Turing machine* R^A *(with access to* A*) such if adversary* A *breaks the combiner then* R^A *breaks the candidate.*

If there is a black-box combiner from a set of tuples of input implementations S to a set of tuples of output implementations S', this is a black-box reduction $S \overset{\text{BB}}{\Longrightarrow} S'$.

In the special case of interactive primitives, the functionality of the primitive can be divided into two parts:

(i) The *next message* function, which determines the next message to be sent by a party (given its partial view of the interaction).

[1] running in polynomial time

(ii) An *output* function, which determines a party's local output (given the view of the entire interaction).

A protocol is then obtained by letting each of the sides alternatively generate their next message by applying the corresponding function to their own local inputs, randomness, and partial view (up to that point in the interaction). Within *transparent* combiners, we allow the use of the primitive only in the context of the protocol (rather than allowing offline access to its oracles).

Definition 12 (Transparent Black-Box Combiner [HKN+05])**.** *A* transparent black-box combiner *is a black-box combiner for an interactive primitive, where every call to a candidate's next message function is followed by this message sent to the other party.*

If there is a transparent black-box combiner from a set of tuples of input implementations S to a set of tuples of output implementations S', this is a transparent black-box reduction $S \overset{\text{TB}}{\Longrightarrow} S'$.

Transparent combiners define something between "normal" black-box and *third party* combiners.

Definition 13 (Third Party Black-Box Combiner [HKN+05])**.** *A* third party black-box combiner *is a black-box combiner where the candidates behave like trusted third parties. The candidates give no transcript to the players, but rather take their inputs and return outputs.*

If there is a third party combiner from a set of tuples of input implementations S to a set of tuples of output implementations S', this is a third party reduction $S \overset{\text{3P}}{\Longrightarrow} S'$.

A third party black-box reduction is harder to achieve than a transparent black-box, a black-box, or even an unconstrained reduction. Therefore, if $S \overset{\text{3P}}{\Longrightarrow} S'$ then $S \overset{\text{TB}}{\Longrightarrow} S'$, if $S \overset{\text{TB}}{\Longrightarrow} S'$ then $S \overset{\text{BB}}{\Longrightarrow} S'$, and if $S \overset{\text{BB}}{\Longrightarrow} S'$ then $S \Longrightarrow S'$.

Furthermore, if $\{C_1, C_2, \ldots\} \Longrightarrow S'$ then $\{C_2, \ldots\} \Longrightarrow S'$, because the combiner guarantees S' for all input configurations C_i (especially, the combiner has to work for the worst-case). The same relation holds for $\overset{\text{3P}}{\Longrightarrow}$, $\overset{\text{TB}}{\Longrightarrow}$, and $\overset{\text{BB}}{\Longrightarrow}$.

In addition to the *k out of n* requirement for the input primitives (as defined in [HKN+05]), [MPW06] proposes more general definitions, which allow for stronger and weaker combiners.

Definition 14 ($(\alpha, \beta; n)$-robust combiner [MPW06])**.** *Let* $\mathbf{T}$ *be a cryptographic primitive for two parties Alice and Bob. An* $(\alpha, \beta; n)$-robust combiner *for* $\mathbf{T}$ *is a probabilistic polynomial time Turing machine that gets* n *candidate schemes implementing* $\mathbf{T}$ *as inputs and implements* $\mathbf{T}$ *while satisfying the following two properties:*

i. If at least α *candidates implement* $\mathbf{T}$ *securely for Alice and at least* β *candidates implement* $\mathbf{T}$ *securely for Bob then the combiner securely implements* $\mathbf{T}$*.*

ii. The running time of the combiner is polynomial in the security parameter m*, in* n*, and in the lengths of the inputs to* $\mathbf{T}$*.*

Definition 15 ($\{\delta, n\}$-robust uniform combiner [MPW06])**.** *A combiner for a two-party primitive* $\mathbf{T}$ *is* $\{\delta, n\}$-robust uniform *if it is an* $(\alpha, \beta; n)$*-robust combiner for* $\mathbf{T}$ *for all* α, β *satisfying* $\alpha + \beta \geq \delta$*.*

3 On combiners for commitment schemes

Within combiners insecure implementations can be broken completely. In practice, a reasonable assumption is to require combiners to ensure tolerance for computational assumptions only. We distinguish two possible interpretations for insecure implementations given as input to a combiner for bit commitment.

3.1 Information theoretic properties are guaranteed

$\mathcal{P}(\mathtt{bc}^{Bh}, \mathtt{bc}^{B}) \overset{\text{3P}}{\Longrightarrow} \{\mathtt{bc}^{Bh}\}$ and $\mathcal{P}(\mathtt{bc}^{H}, \mathtt{bc}^{bH}) \overset{\text{3P}}{\Longrightarrow} \{\mathtt{bc}^{bH}\}$
but $\{(\mathtt{bc}^{Bh}, \mathtt{bc}^{H}), (\mathtt{bc}^{B}, \mathtt{bc}^{bH})\} \overset{\text{TB}}{\not\Longrightarrow} \mathbf{BC}$

On one hand we could say that only the cryptographic property relies on a computational assumption, and that the information theoretic property of a commitment scheme (binding or hiding) still holds, even for insecure implementations. These input scenarios often can be combined trivially.

For two information theoretic binding implementations (binding is guaranteed for both input implementations), i.e., the input set is $\mathcal{P}(\mathtt{bc}^{Bh}, \mathtt{bc}^{B}) = \{(\mathtt{bc}^{Bh}, \mathtt{bc}^{B}), (\mathtt{bc}^{B}, \mathtt{bc}^{Bh})\}$, a $(1;2)$-robust combiner can be constructed, protecting the sender against incorrect cryptographic assumptions for hiding.

Theorem 1. *There exists a third party black-box $(1;2)$-robust combiner for bit commitment with guaranteed binding property of the candidate schemes.*

Proof. Alice commits to b_1 and b_2 with one scheme, respectively, where $b = b_1 \oplus b_2$. If one scheme leaks the committed bit, i.e., the hiding property is violated, the receiver still does not obtain any information about b. □

Corollary 1. *There exists a third party black-box $(2,1;2)$-robust combiner for bit commitment.*

The same holds for the hiding property with input set $\mathcal{P}(\mathtt{bc}^{bH}, \mathtt{bc}^{H})$.

Theorem 2. *There exists a third party black-box $(1;2)$-robust combiner for bit commitment with guaranteed hiding property of the candidate schemes.*

Proof. Alice commits to the same value twice. If both openings lead to the same bit, i.e., $b'_1 = b'_2$, Bob accepts. Alice was not able to cheat, because at least one candidate scheme is secure. Since both schemes are hiding, Bob can only flip a coin to guess the original bit before the opening phase. □

Corollary 2. *There exists a third party black-box $(1,2;2)$-robust combiner for bit commitment.*

It seems hard to construct a combiner from a hiding and a binding scheme, i.e., possible input implementations are $(\mathtt{bc}^{Bh}, \mathtt{bc}^{H})$ and $(\mathtt{bc}^{B}, \mathtt{bc}^{bH})$.

Definition 16. *A* straightforward combiner for commitment schemes *is a transparent black-box combiner with the following properties: After the commit phase, Alice has commited to a sequence of bits B. The open procedure consists of Alice opening the commitments and optionally sending an additional value ρ_s. Then, Bob efficiently computes the bit (b') and decides whether he accepts (if $a' = \top$), i.e., Bob computes $f(B', \rho_s) = (b', a')$.*

Theorem 3. *There is no straightforward $(1;2)$-robust combiner for bit commitment if one candidate scheme is information theoretic binding, the other candidate scheme is information theoretic hiding, and at least one scheme's computational assumption is fulfilled.*

Proof. We prove by contradiction. We assume to have a straightforward $(1;2)$-robust combiner in a world with **PSPACE**-complete oracle. Alice commits to bits $B = (b_1, b_2, b_3, \ldots)$. The original bit b and a truth value a whether Bob accepts the commitment can be computed from B, i.e., $(b, a) = f(B, \rho_s)$. In the decommit phase, Alice opens these bits and Bob gets $B' = (b'_1, b'_2, b'_3, \ldots)$. Bob can efficiently compute the value $(b', a') = f(B', \rho_s)$ and he accepts if $a' = \top$. A combiner using only one scheme might rely on the wrong one, therefore, we only consider combiners using both schemes. We split the input bits into bits committed with the binding scheme, namely $B^B = (b_1^B, b_2^B, b_3^B, \ldots)$, and bits of the hiding scheme, namely $B^H = (b_1^H, b_2^H, b_3^H, \ldots)$. After the commit protocol of the combiner, Bob has received commitments and messages of both schemes, say $C^B = (c_1^B, c_2^B, c_3^B, \ldots), \bar{M}^B = (\bar{m}_1^B, \bar{m}_2^B, \bar{m}_3^B, \ldots)$, and $C^H = (c_1^H, c_2^H, c_3^H, \ldots), \bar{M}^H = (\bar{m}_1^H, \bar{m}_2^H, \bar{m}_3^H, \ldots)$. If the binding scheme $\mathtt{bc}^B$ was insecure for Alice, Bob could open all bits committed with it, i.e., he can compute efficiently $B^B = (b_1^B, b_2^B, b_3^B, \ldots)$ from C^B and $\bar{M}^B$. Since the combiner is assumed secure, Bob must not be able to compute efficiently the original bit b, i.e., there is no efficient algorithm A' computing b given B^B (and $C^H, \bar{M}^H$, but without B^H).

Since the combiner is running in polynomial time, f has to be computable in polynomial time. Bob asks the **PSPACE**-complete oracle to count the size of two sets, namely $r_0 := |\{(\tilde{B}^H, \tilde{r}) : f(B^B \tilde{B}^H, \tilde{r}) = (0, \top)\}|$ and $r_1 := |\{(\tilde{B}^H, \tilde{r}) : f(B^B \tilde{B}^H, \tilde{r}) = (1, \top)\}|$. If $r_0 \not\approx r_1$ then the commitment is not hiding and therefore, Bob can predict b. If both $r_0 > 0$ and $r_1 > 0$, Alice is able to cheat efficiently if the hiding scheme is not binding (simply by asking the **PSPACE**-complete oracle for two configurations (B^H, ρ_s) and (B'^H, ρ'_s) satisfying $f(B^B B^H, \rho_s) = (0, \top)$ and $f(B^B B'^H, \rho'_s) = (1, \top)$). She

can choose B^B as before, then cheat about B^H (since the second scheme is not binding in this case), and therefore, open the commitment for two different values efficiently. □

An analogue proof works for "straightforward" combiners for **OT** (see Definition 20).

Another proof for the theorem is similar to the impossibility proof for **OT** of [HKN+05]. The crucial point is the definition of the **BC** oracles used. From this theorem we also conclude Corollary 3 and 4.

Theorem 4. *There is no transparent black-box* $(1;2)$*-robust combiner for bit commitment if one candidate scheme is information theoretic binding, the other candidate scheme is information theoretic hiding, and at least one scheme's computational assumption is fulfilled.*

Proof. The proof is similar to the one of [HKN+05, Theorem 4.1]. Usually, black-box separation proofs ($A \not\Rightarrow B$) work by constructing a world (W) wherein A exists but B does not. Since $A \Rightarrow B$ would have to hold in any world (including W), this argument suffices. The standard approach is not possible for a $(1;2)$-robust combiner for **T**, because if an implementation $\mathtt{t}$ exists, there is a trivial combiner simply using $\mathtt{t}$. Therefore, one method for an impossibility proof for combiners is to construct two worlds $(\mathsf{World}_1, \mathsf{World}_2)$ wherein **T** exists, but any combiner is secure in at most one of these worlds.

The proof is by contradiction. We assume there exists a transparent black-box $(1;2)$-robust combiner for **BC**, say $\mathtt{bc}_{cmb}(\cdot,\cdot)$. In any world where **BC** exists, the combiner has to work as specified, i.e., if at most one computational assumption is broken, the combiner is still secure. We show two worlds $(\mathsf{World}_1, \mathsf{World}_2)$ wherein **BC** exists such that every transparent black-box combiner for **BC** is insecure in at least one of them.

Before constructing the worlds, we determine implementations of **BC** to be placed in the worlds. The implementations make use of two oracles $(\mathcal{O}^{Bh}, \mathcal{O}^{bH})$ defined next. The first implementation, $\mathtt{bc}^{Bh}$, is information theoretic binding and consists of an oracle implementing a random permutation $(\mathcal{O}^{Bh})$. The second implementation, $\mathtt{bc}^{bH}$, is information theoretic hiding. For every commitment c there are exactly two possibilities to open it, one for $b = 0$ and one for $b = 1$.

$$\begin{aligned}
\mathcal{O}^{Bh} &: \{0,1\} \times \{0,1\}^n \to \{0,1\}^{n+1} \text{ random permutation} \\
\mathcal{O}^{bH} &: \{0,1\} \times \{0,1\}^n \to \{0,1\}^n \text{ random function, such that} \\
&\quad \forall c \in \{0,1\}^n \exists \rho_0, \rho_1 \in \{0,1\}^n : \mathcal{O}^{bH}(0,\rho_0) = \mathcal{O}^{bH}(1,\rho_1) = c
\end{aligned}$$

The protocols make use of these oracles. In the COMMIT phase of both protocols $(\mathtt{bc}^{Bh}, \mathtt{bc}^{bH})$, the sender chooses bit b and randomness ρ_s and queries the oracle, $c = \mathcal{O}(b, \rho_s)$. The resulting commitment (c) is sent to the receiver. In the OPEN phase, the sender reveals his choices (b, ρ_s) and the receiver checks if the values are consistent with c by querying the corresponding oracle.

Claim. *Both oracles enable the execution of a* **BC** *protocol even in the presence of a* **PSPACE***-complete oracle.*

For both implementations the **PSPACE**-complete oracle does not help. To compute the original bit in the binding implementation $(\mathtt{bc}^{Bh})$, all the receiver can do is to query the oracle in an exhaustive search (the expected number of queries needed is 2^{n-1}). Finding a collision in the hiding implementation $(\mathtt{bc}^{bH})$ is possible only by excessively querying the oracle $\mathcal{O}^{bH}$ (according to the birthday paradox, the expected number of queries needed to find a collision is $\sqrt{2^n}$). ∎

On the assumption that $\mathtt{bc}_{cmb}(\cdot,\cdot)$ is a secure combiner, $\mathtt{bc}_{cmb}(\mathtt{bc}^{Bh}, \mathtt{bc}^{H})$ and $\mathtt{bc}_{cmb}(\mathtt{bc}^{B}, \mathtt{bc}^{bH})$ are secure implementations of **BC** in any world.

The combiner's task is to ensure tolerance if one implementation is insecure. We construct inverters to selectively break one implementation's security. The binding oracle is made flawed by an inverter $(\mathcal{O}^{Bh})^{-1}$ computing the committed bit and the randomness used. The hiding oracle is made flawed by an oracle $(\mathcal{O}^{bH})^{-1}$ computing r_0 and r_1 given c.

We now construct the two worlds. In each world one of the implementations is made flawed by adding the inverter.
World_1 contains a **PSPACE**-complete oracle, $\mathcal{O}^{Bh}$, $\mathcal{O}^{bH}$, and $(\mathcal{O}^{Bh})^{-1}$.
World_2 contains a **PSPACE**-complete oracle, $\mathcal{O}^{Bh}$, $\mathcal{O}^{bH}$, and $(\mathcal{O}^{bH})^{-1}$.

Furthermore, we construct an "insecure" BareWorld, which is later compared with World_1 and World_2. The BareWorld contains a **PSPACE**-complete oracle only. The **BC** oracles are simulated with naive implementations. In the simulation of an oracle $\mathcal{O}$, the simulating party receives bit and randomness from the committing party, answers with a random value that preserves the oracle's conditions, and saves the tuple in the function table for future queries, i.e.:

- Within $\mathtt{bc}^{Bh}$, the receiver simulates the oracle $\mathcal{O}^{Bh}$, i.e., for any committed bit, he receives bit and randomness from the committing party and sends back a random value that preserves the permutation

condition (and saves the tuple for future queries, i.e., he maintains a function table). The resulting (insecure) oracle simulation is denoted by $\mathcal{R}(\mathcal{O}^{Bh})$.

- The sender simulates the oracle $\mathcal{O}^{bH}$ within the hiding scheme ($\mathtt{bc}^{bH}$), i.e., for any committed bit, he receives bit and randomness from the committing party (which usually is himself) and sends back a random n-bit string that preserves the collision condition (and saves the tuple). The resulting (insecure) oracle simulation is denoted by $\mathcal{S}(\mathcal{O}^{bH})$.

Claim. *The combiner in the* BareWorld *is insecure.*

The simulations are insecure. $\mathcal{R}(\mathcal{O}^{Bh})$ is not hiding and $\mathcal{S}(\mathcal{O}^{BH})$ is not binding. **BC** does not exist without any hardness assumption. The combiner cannot build a commitment scheme from scratch in the presence of a **PSPACE**-complete oracle. Therefore, the resulting implementation $\mathtt{bc}_{cmb}(\mathcal{R}(\mathcal{O}^{Bh}), \mathcal{S}(\mathcal{O}^{bH}))$ cannot implement a secure commitment scheme. Thus, there is a successful attack in the BareWorld. ∎

Claim. *Every successful attack in the* BareWorld *results in an attack in* World_1 *or in* World_2.

For an illustration, see Figure 1.

If the receiver in the BareWorld is able to violate the hiding property, then the receiver in World_1 is able to compute the bit, because he has access to the same information, namely to all commitments of $\mathtt{bc}^{bH}$ and all bits, random values, and commitments of $\mathtt{bc}^{Bh}$. In addition, in the BareWorld, the implementation of $\mathtt{bc}^{Bh}$ is not binding for the receiver, because he simulates $\mathcal{O}^{Bh}$ himself. However, the capability to open his committed bits arbitrarily does not help the receiver to guess the sender's bit.

If the sender in the BareWorld is able to violate binding, then the sender in World_2 is able to violate the binding property, because he can influence the same values, namely the commitments of $\mathtt{bc}^{bH}$. In addition, in the BareWorld, the implementation of $\mathtt{bc}^{bH}$ is not hiding for the receiver, because the sender simulates $\mathcal{O}^{bH}$ himself. However, the capability to open the receiver's committed bits does not help the sender to violate the binding property. ∎

Any of these attacks contradicts the assumption. Thus, there is no such combiner. □

A $\{3,2\}$-robust uniform combiner for **BC** has to work for $\mathcal{P}(\mathtt{bc}^{bh}, \mathtt{bc}^{b}) \cup \mathcal{P}(\mathtt{bc}^{bh}, \mathtt{bc}^{h})$, where, compared to the inputs in Theorem 4, $\{(\mathtt{bc}^{bH}, \mathtt{bc}^{B}), (\mathtt{bc}^{H}, \mathtt{bc}^{Bh})\}$, even fewer security properties are guaranteed. This observation results in the following corollary.

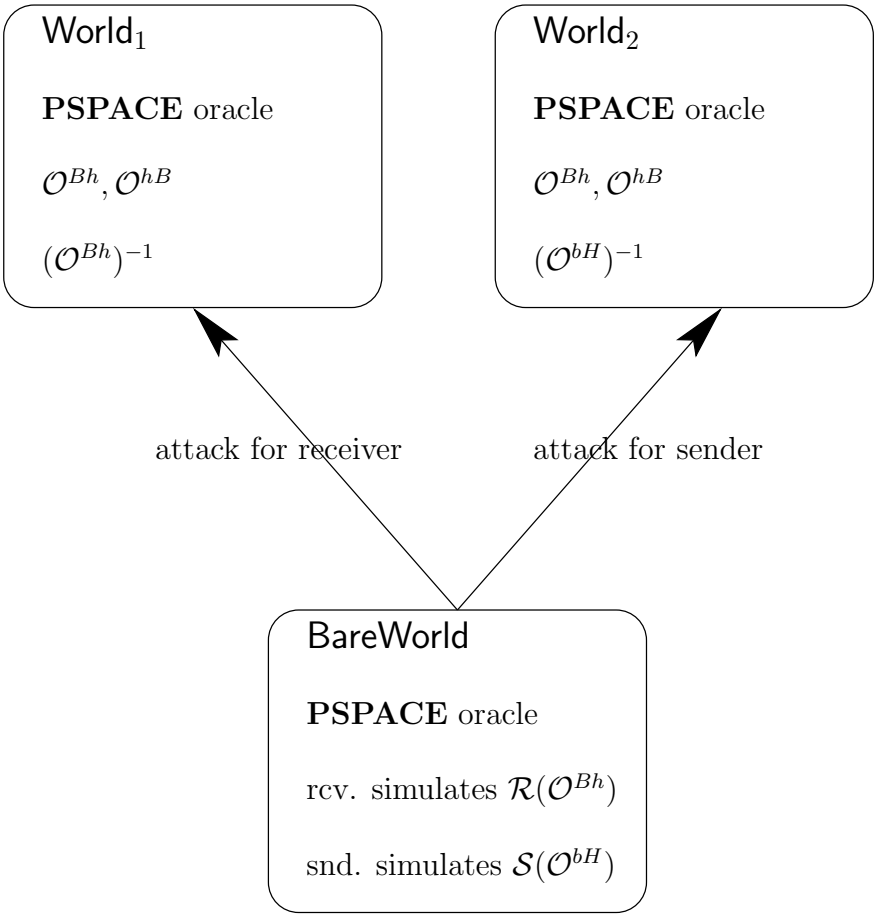

Figure 1: Illustration for the proof of Theorem 4

Corollary 3. *There is no transparent black-box* $\{3,2\}$*-robust uniform combiner for bit commitment.*

3.2 No restriction on candidate schemes

3.2.1 $(1;2)$-robust combiners

$\mathcal{P}(\mathsf{bc}^{bh}, \mathsf{bc}) \overset{\text{TB}}{\not\Longrightarrow} \mathbf{BC}$ but $\mathcal{P}(\mathsf{bc}^{bh}, \mathsf{bc}) \overset{\text{BB}}{\Longrightarrow} \{\mathsf{bc}^{Bh}\} = \mathbf{BC}$

In the previous section, information theoretic properties were guaranteed. Alternatively, we could say that an insecure implementation only fulfils the specified functionality but no security can be assumed at all. This

corresponds to the definition of combiners (Definition 10). In this case, the construction of a combiner is more difficult.

A $(1;2)$-robust combiner for **BC** has to work for $\mathcal{P}(\mathtt{bc}^{bh}, \mathtt{bc})$, where, compared to the inputs in Theorem 4, $\{(\mathtt{bc}^{bH}, \mathtt{bc}^{B}), (\mathtt{bc}^{H}, \mathtt{bc}^{Bh})\}$, even fewer security properties are guaranteed. This observation results in the following corollary.

Corollary 4. *There is no transparent black-box* $(1;2)$*-robust combiner for bit commitment.*

Combiners are required to run in polynomial time. Furthermore, [Her05] pointed out that combiners, for their use in practice, should be "efficient" (e.g., only little additional computation and only few calls of the input implementations necessary). The black-box construction in this section is of theoretical interest. Even though the construction runs in polynomial time the reductions used within are too slow for practical use.

Using the reductions from commitment schemes to one-way functions [IL89], from one-way functions to pseudo-random generators [HILL99], and from pseudo-random generators to binding commitment schemes [Nao91], a non-transparent black-box combiner for commitment schemes can be obtained.

Theorem 5. *There exists a black-box* $(1;2)$*-robust combiner for* **BC**.

Proof. The combiner is constructed as follows (cf. Figure 2).

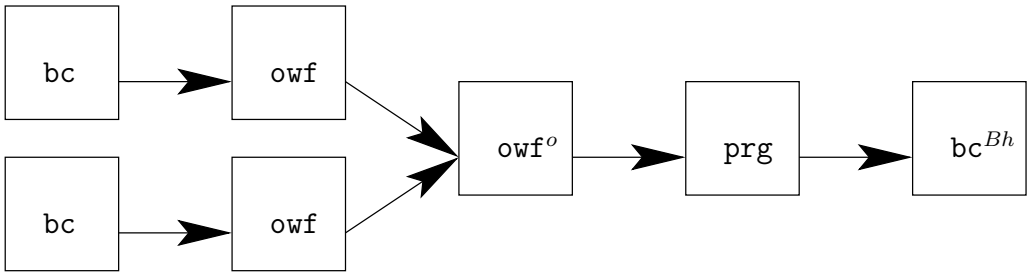

Figure 2: Illustration for the four step reduction in the proof of Theorem 5

(i) **BC** $\Longrightarrow$ **OWF**: From both commitment schemes given as input, a one-way function can be obtained [IL89]. One of these one-way functions is secure because one of the commitment schemes is secure.

(ii) $(1;2)$-robust combiner for **OWF** ($\mathcal{P}(\mathtt{owf}^o, \mathtt{owf}) \Longrightarrow$ **OWF**): In the next step, the combiner for one-way functions is used, i.e., $f(x|y) =$

$f_1(x)|f_2(y)$ (where | denotes string concatenation). The resulting function (f) is one-way if at least one of the input functions (f_1, f_2) is one-way [Her05].

(iii) **OWF** $\Longrightarrow$ **PRG**: The one-way function is used to obtain a pseudo-random generator [HILL99].

(iv) **PRG** $\Longrightarrow$ **BC**: The pseudo-random generator is used to construct a statistically binding commitment scheme [Nao91]. □

This construction cannot be applied within a transparent black-box combiner. The one-way function is obtained by a simulation of the commitment scheme. A transparent black-box combiner is obliged to send these messages to the other party. The constructions of [Nao91, HILL99] need offline access to the **OWF**.

To illustrate how we can construct a one-way function from a bit commitment scheme, we show the construction from statistically binding commitment schemes to one-way functions. For the general case we refer to [IL89].

Lemma 1. *There exists a black-box reduction from statistically binding bit commitment schemes to one-way functions.*

Proof. We follow the guideline of [Gol00, Exercise 13, Page 326]. A protocol consists of two probabilistic polynomial time Turing machines, the randomness is given as random bits on an additional tape. Within the commit protocol, the sender uses randomness ρ_s to commit to a bit b and the receiver uses randomness ρ_r. The communication of the commit protocol is a sequence $\bar{m}$ of messages.

We consider the mapping to the receiver's view, $(b, \rho_s, \rho_r) \mapsto (\rho_r, \bar{m})$, containing his random bits and all messages.

By Definition 1, a one-way function needs to be easy to compute and hard to invert. The first requirement is guaranteed by construction. Since the Turing machines used in the bit commitment scheme run in polynomial time, we can simulate the sender's machine S and the receiver's machine R in polynomial time to obtain the output $f(b|\rho_s|\rho_r) = (r|\bar{m})$. The second requirement is proven by contradiction. We assume that there is an algorithm A' that computes an inverse with non-negligible probability, i.e., $A'(\rho_r|\bar{m}) = (b'|\rho'_s|\rho'_r)$ with non-negligible probability.

- If $\rho'_r \neq \rho_r$, A' did not compute an inverse, because, by construction of f, all function pairs are of the form $(b|\rho_s|\rho_r, \rho_r|\bar{m}) \in f$.
- If $\rho'_r = \rho_r$, two cases are distinguished:

– Using the statistical unambiguity requirement of the commitment scheme, $b' \neq b$ is possible only for a negligible fraction of the coin tosses of the receiver (ρ_r) and therefore, only for a negligible fraction of the $\rho_r|\bar{m}$.
– If $b' = b$, we may use A' to break the commitment's secrecy; however, $\{\langle S(0), R^* \rangle (1^n)\}_{n\in\mathbb{N}}$ and $\{\langle S(1), R^* \rangle (1^n)\}_{n\in\mathbb{N}}$ are required to be computationally indistinguishable.

The computational assumption for the hiding property results in the computational assumption that f is prevented from being inverted. □

3.2.2 Combiners for a secure majority

$\mathcal{P}(\mathtt{bc}^{bh}, \mathtt{bc}^{bh}, \mathtt{bc}) \cup \mathcal{P}(\mathtt{bc}^{bh}, \mathtt{bc}^{b}, \mathtt{bc}^{h}) \overset{\text{3P}}{\Longrightarrow} \mathbf{BC}$

$(\lfloor \frac{n}{2} \rfloor + 1; n)$-robust combiners for **BC** were presented in [Her05] using a t-out-of-n secret sharing [Sha79] for a threshold t. $t+1$ shares uniquely determine the shared value v, but t shares do not give any information about v. Sharing combiners are defined as follows: given a secret sharing scheme $\langle \textsc{Share}, \textsc{Reconstruct} \rangle$, Alice generates the shares for bit b using randomness ρ_s and she then commits with one scheme per share, i.e., $\mathtt{bc}_i.\textsc{Commit}(\textsc{Share}_i(b, \rho_s))$. In the Open phase, she opens all commitments and Bob applies $b = \textsc{Reconstruct}(b_1 \dots b_n)$. If the shares are consistent, he accepts b. A security proof is given within the proof of Theorem 7.

In general, any $\lfloor \frac{n}{2} \rfloor + 1$ implementations can be secure for the receiver (binding) and any (maybe other) $\lfloor \frac{n}{2} \rfloor + 1$ candidates can be secure for the sender (hiding). Since $2 \cdot (\lfloor \frac{n}{2} \rfloor + 1) > n$, at least one implementation is both binding and hiding. This is called a $(\lfloor \frac{n}{2} \rfloor + 1, \lfloor \frac{n}{2} \rfloor + 1; n)$-robust combiner (Definition 14). We give a concrete implementation of a $(2, 2; 3)$-robust combiner, which also implies a $(2; 3)$-robust combiner.

Theorem 6. *There exists a third party black-box $(2, 2; 3)$-robust combiner for bit commitment.*

Proof. Bob is protected against a wrong binding assumption using redundancy and Alice is protected against a wrong hiding assumption using $\oplus$. Alice commits to random b_1, b_2, and b_3 satisfying $b = b_1 \oplus b_2 \oplus b_3$. With each implementation Alice commits to two values as defined in Figure 3 (a share consists of two bits). If one implementation $\mathtt{bc}_i$ is insecure for Bob, i.e., not binding, he will catch a malicious Alice using the redundancy, because Alice has committed to both bits with the other schemes.

	b_1	b_2	b_3
$\mathtt{bc}_1$	–	✓	✓
$\mathtt{bc}_2$	✓	–	✓
$\mathtt{bc}_3$	✓	✓	–

Figure 3: With each input implementation of the $(2, 2; 3)$-robust combiner Alice commits to two values (a ✓ denotes a commitment for this bit), e.g., with $\mathtt{bc}_1$ Alice commits to b_2 and to b_3.

With one implementation $\mathtt{bc}_i$ violating the hiding property, Bob does not learn the third value b_i and therefore, Bob does not obtain any information about b. □

Corollary 5. *There exists a third party black-box* $(2; 3)$*-robust combiner for bit commitment.*

Note that the combiners with guaranteed binding and guaranteed hiding (Corollary 1 and 2) are special cases of the sharing combiner.

Theorem 7. *There exists a third party black-box* $(\alpha, \beta; n)$*-robust combiner for* **BC***, if and only if* $\alpha + \beta > n$*.*

Proof. If $\alpha + \beta \leq n$, the set of tuples of input implementations might consist of α trivial simulations of a hiding commitment (Alice simulates a random oracle, not binding) and β trivial simulations of a binding commitment (Bob simulates a random oracle, not hiding), i.e., two distinct ensembles, one with the binding schemes and one with the hiding schemes. No implementation can provide a secure commitment for both parties. There is no commitment scheme without any hardness assumption. Thus, there is no combiner that is secure for these input implementations.

If $\alpha + \beta > n$, using the sharing combiner of [Her05], a robust combiner is constructed. The sharing has degree $n - \alpha$, therefore, $n - \alpha$ commitments with broken hiding property do not leak any information about the original bit. The degree is at most $\beta - 1$, $(\beta > n - \alpha)$. At least β implementations are secure binding. Therefore, Bob will catch a malicious Alice using the redundancy. □

This characteristic of **BC** is very similar to **OT**. Since **OT** is symmetric[2] [WW04], there are more tolerant combiner constructions [MPW06]. In contrast, **BC** is not known to be symmetric.

[2]It is possible to use an **OT**-protocol with sender A and receiver B to simulate an **OT**-protocol with swapped roles as follows: A sets $c_{sim} = b_0 \oplus b_1$ and receives $b_{c_{sim}}$, B chooses ρ at random and sets $b_{0,sim} = \rho$, $b_{1,sim} = \rho \oplus c$ (i.e., $\mathtt{ot}(\rho, \rho \oplus c; b_0 \oplus b_1)$). Afterwards, A sends $m = b_0 \oplus b_{c_{sim}}$. Then, B can compute $b_c = \rho \oplus m$.

3.3 Summary

We briefly summarise our contribution and known results in the following table using the notation introduced in this thesis.

$\mathcal{P}(\mathtt{bc}^{Bh}, \mathtt{bc}^{B}) \overset{\text{3P}}{\Longrightarrow} \{\mathtt{bc}^{Bh}\} = \mathbf{BC}$	Theorem 1, Corollary 1
$\mathcal{P}(\mathtt{bc}^{bH}, \mathtt{bc}^{H}) \overset{\text{3P}}{\Longrightarrow} \{\mathtt{bc}^{bH}\} = \mathbf{BC}$	Theorem 2, Corollary 2
$\{(\mathtt{bc}^{bH}, \mathtt{bc}^{B}), (\mathtt{bc}^{H}, \mathtt{bc}^{Bh})\} \overset{\text{TB}}{\not\Longrightarrow} \mathbf{BC}$	Theorem 4
$\mathcal{P}(\mathtt{bc}^{bh}, \mathtt{bc}) \overset{\text{TB}}{\not\Longrightarrow} \mathbf{BC}$	Corollary 4
$\mathcal{P}(\mathtt{bc}^{bh}, \mathtt{bc}) \overset{\text{BB}}{\Longrightarrow} \{\mathtt{bc}^{Bh}\} = \mathbf{BC}$	[IL89, Nao91, HILL99, Her05] (Theorem 5)
$\mathcal{P}(\mathtt{bc}^{bh}, \mathtt{bc}^{bh}, \mathtt{bc}) \overset{\text{3P}}{\Longrightarrow} \{\mathtt{bc}^{bh}\} = \mathbf{BC}$	[Her05], Corollary 5
$\mathcal{P}(\mathtt{bc}^{bh}, \mathtt{bc}^{bh}, \mathtt{bc}) \cup \mathcal{P}(\mathtt{bc}^{bh}, \mathtt{bc}^{b}, \mathtt{bc}^{h}) \overset{\text{3P}}{\Longrightarrow} \mathbf{BC}$	Theorem 6

4 On combiners for interactive proof systems

A $(1;2)$-robust combiner for zero-knowledge interactive proof systems gets as input 2 interactive proof systems $(\mathtt{ip}_1, \mathtt{ip}_2)$ where one of the input systems can be insecure. Insecure means that the system could fail on three counts, namely on soundness, completeness, and on the zero-knowledge property.

Definition 17. *A* straightforward $(k;n)$-robust combiner for interactive proof systems for a language L *is a transparent black-box* $(k;n)$*-robust combiner that performs the following steps (applied on a word* $x \in L$*, using the input proof systems* $\mathtt{ip}_1 \dots \mathtt{ip}_n$*):*

i. Using a split-up-function *F, n words $x_1 \dots x_n$ are generated from x.*

ii. $x_i \in L$ is proven using $\mathtt{ip}_i$.

iii. The verifier accepts if all proofs were successful.

In the following we distinguish different restrictions of a potential failure.

4.1 Interactive proof systems without zero-knowledge

We start by dropping the zero-knowledge property. Thus, we analyse interactive proof systems only. Usually, candidate schemes are tested for their functionality. However, computational assumptions could be broken. Therefore, a reasonable scenario would be to assume guaranteed completeness for both input candidates $\mathtt{ip}_i$. Robust combiners for interactive proof systems with guaranteed completeness and without zero-knowledge property are easy to construct.

Theorem 8. *There exists a third party black-box* $(1;2)$*-robust combiner for interactive proof systems with guaranteed completeness of the candidate systems.*

Proof. The combiner's task is to ensure soundness. The input word $x \in L$ is proven with both input protocols $(\mathtt{ip}_1, \mathtt{ip}_2)$. The verifier accepts if and only if both proofs are correct (i.e., the verifier is convinced in both protocols). One of the input protocols is sound and therefore, the resulting combination is sound. □

Apart from being quite unrealistic, soundness might be guaranteed, requiring the combiner to ensure completeness.

Theorem 9. *There exists a third party black-box* $(1;2)$*-robust combiner for interactive proof systems with guaranteed soundness of the candidate systems.*

Proof. The combiner's task is to ensure completeness. The input word $x \in L$ is proven with both input protocols $(\mathtt{ip}_1, \mathtt{ip}_2)$. The verifier accepts if at least one of the proofs is correct. One of the input protocols is complete and therefore, the combination is complete. □

If neither completeness nor soundness is guaranteed for the broken scheme, it seems hard to construct a transparent black-box $(1;2)$-robust combiner.

Theorem 10. *There is no third party black-box* $(1;2)$*-robust combiner for interactive proof systems if neither completeness nor soundness is guaranteed for the broken scheme.*

Proof. Within any third party black-box combiner for interactive proof systems a number of statements are proven with the candidate schemes $\mathtt{ip}_i$ and in the end, the verifier has to decide whether he accepts the proof. If accepting proofs of both $\mathtt{ip}_1$ and $\mathtt{ip}_2$ are needed to convince the verifier, the prover is possibly unsuccessful if one candidate is not complete. If the verifier accepts proofs of one scheme, this candidate might not be sound, and therefore, the prover can be able to construct an accepting proof. □

If no assumption apart from functionality can be made for insecure candidates, we can still construct a robust combiner given that the majority of the input implementations are secure.

Theorem 11. *There exists a third party black-box* $(2;3)$*-robust combiner for interactive proof systems.*

Proof. The combiner's task is to ensure both soundness and completeness. The input word $x \in L$ is proven with all input protocols ($\mathtt{ip}_1$, $\mathtt{ip}_2$, and $\mathtt{ip}_3$). The verifier accepts if at least two of the proofs are correct. Two of the input protocols are sound and complete and therefore, the combiner is sound and complete. □

When taking the zero-knowledge property into the scope, combiners are not that easy to obtain.

4.2 Zero-knowledge combiners

Theorem 10 implies the following corollary.

Corollary 6. *There is no third party black-box* $(1;2)$*-robust combiner for zero-knowledge proof systems.*

Therefore, if we consider third party black-box $(1;2)$-robust combiners for interactive proof systems, either soundness or completeness must be guaranteed. It is reasonable to assume completeness.

To obtain the zero-knowledge property for the combiner, a simulator for the whole transcript is needed. Only simulators for secure input protocols are guaranteed to exist. The simulator must be able to generate transcripts of the insecure protocols even if they leak the witness of $x_i \in L$. It is unclear how a simulator for a protocol with broken zero-knowledge property should behave, if it even exists.

The following lemma shall give initial intuition for the problem of combiners for interactive proof systems.

Lemma 2. *Within a straightforward combiner for zero-knowledge interactive proof systems applied for a word* x*, no output of the split-up-function may be equal to* x*.*

Proof. If $\mathtt{ip}_j$ is insecure for the prover, i.e., the zero-knowledge property is not guaranteed for $\mathtt{ip}_j$, in the worst-case, $\mathtt{ip}_j$ writes the witness for $x \in L$ within the transcript. No efficient simulator can generate such a transcript. □

4.3 Witness-hiding combiners with guaranteed soundness and completeness of the candidate systems

Definition 18. *A* split reduction for a language $L \in \mathbf{NP}$ *consists of two functions (computable in polynomial time)* $F : x \mapsto (x_1, x_2)$ *and* $F' : (x, w) \mapsto (w_1, w_2)$*.* $x \in L$ *denotes the word and* w *its witness. The functions satisfy the following properties;*

- $x \in L \Leftrightarrow x_1, x_2 \in L$*,*
- w_i *is the witness for* $x_i \in L$*, and*
- w_i *does not give any information about* w*.*

Lemma 3. *If there is a split reduction for a language* $L \in \mathbf{NP}$ *then there is a* $(1;2)$*-robust combiner for witness-hiding interactive proof systems with guaranteed soundness and completeness of the candidate systems.*

Proof. Three properties need to be verified:

- Completeness and soundness: both input proofs $\mathtt{ip}_i$ are complete and sound, therefore, by construction, the combination is complete and sound.

- Witness-hiding: Since w_i does not give any information about w (construction of the split reduction) and since at most one input proof is not witness-hiding, the verifier is unable to obtain w from one w_i. □

If $\mathbf{P} \neq \mathbf{NP}$, the split reduction must not reduce the problem size (word length $|x|$), by a (at least) linear factor. Otherwise, an efficient algorithm could recursively apply this reduction and finally, by using an exponential algorithm on the instances of logarithmic size, compute the result for x. Furthermore, if the new instances (x_1, x_2) depend on the witness, i.e., $F : (x, w) \mapsto (x_1, x_2)$, but there is no efficient function $\tilde{F} : x \mapsto (x_1, x_2)$, then F has to be one-way. We considered reducing an instance of 3-Coloring [GJ90] to an instance of SubgraphIsomorphism and an instance of 3-Coloring by randomly blowing up (and obfuscating) a graph G into a new graph G', then proving that G is a subgraph and that G' is still 3-colorable. However, a secure construction of a graph G' of this type would imply a **OWF**. Having a **OWF** at hand, we could construct **BC** and then use the [BCC88] protocol.

Another problem we considered was SubsetSum [GJ90]. The question is whether any subset of a given set of integers $A = \{a_1, \ldots a_n\}$ sums up to B. A natural way for a combiner would be to (randomly) split A into two distinct subsets (A_1, A_2) with requested subset sum B_1, B_2. A broken proof system would leak only partial information to a malicious verifier; however, this is not enough to reach the zero-knowledge property. Possible modifications include adding or removing elements. Unfortunately, a good compromise between the prover's security and the soundness of the proof scheme is difficult to reach.

4.4 Candidate systems with guaranteed completeness

Theorem 12. *There is no deterministic straightforward transparent black-box* $(1; 2)$*-robust combiner for zero-knowledge interactive proof systems with guaranteed completeness of the candidate systems.*

Proof. L is assumed to be **NP**-complete. We show that such a combiner would provide an efficient solution of $x \in L$. An insecure protocol might leak the witness for any $x_i \in L$ proven using it. Since the simulator has to work for all input protocols and their permutations, it must be able to generate

transcripts that leak the witness for any possible insecure protocol $\mathtt{ip}_j$. The combiner is deterministic and therefore, the x_i generated within are always the same. Every x_i is proven with at least one protocol $\mathtt{ip}_j$ (otherwise we omit generating x_i). The algorithm that breaks the combiner applies the efficient simulator for different insecure protocols $\mathtt{ip}_k$ until transcripts for all x_i are generated (since every x_i is proven with at least one protocol $\mathtt{ip}_j$ and this protocol could be insecure, i.e., $j = k$, we obtain this transcript) and extracts the witness for all transcripts. □

Instead of a deterministic combiner there might be a randomised one where the simulator could prepare a witness for a subset of the x_i when it is generated.

Theorem 13. *There is no straightforward transparent black-box* $(1; 2)$*-robust combiner for zero-knowledge interactive proof systems with guaranteed completeness of the candidate systems.*

Proof. We give an efficient algorithm for the prover to cheat on the verifier. A simulator for the combiner would have to be able to generate a transcript where the witness for all proofs using $\mathtt{ip}_i$ is leaked. If the prover does know which of the $\mathtt{ip}_i$ is insecure (the simulator does not have information about this), the prover can cheat as follows: he applies the simulator until it generates a protocol where the witness for all applications of the secure protocol is leaked. Then, if the insecure protocol is unsound, the prover can cheat about these proofs and at the same time use the witness from the transcript for the secure proofs. □

5 On combiners for oblivious transfer

Definition 19. *A* uni-directional combiner for an interactive primitive **T** *between A and B is a combiner that must not use any input implementation* `t` *in the other direction, i.e., change the role of A and B within* `t`.

Uni-directional combiners are somewhat similar to straightforward combiners (Definition 16 and 20).

Theorem 14. *There is no uni-directional transparent black-box $(1;2)$-robust combiner for oblivious transfer if the privacy of the receiver in the first implementation or the privacy of the sender in the second implementation can be broken.*

Proof. We use the proof of [HKN+05]. They defined oracles (length tripling random functions $\{0,1\}^n \to \{0,1\}^{3n}$) to implement **OT**:

$f_1 : (\rho_r, c) \mapsto m_1$	computes the first message m_1 from the receiver's choice bit c and randomness ρ_r,
$f_2 : (\rho_s, b_0, b_1, m_1) \mapsto m_2$	computes the answer message m_2 from the sender's randomness ρ_s, his two bits b_i, and the first message, and
$R : (m_2, \rho_r, c) \mapsto s_c$	extracts the chosen bit.

In their proof, any implementation (defined with three oracles) is broken completely, i.e., with inverters f_1^{-1} and f_2^{-1}. In this proof, we want to break selectively the privacy of a single party per protocol. We claim that if the role of the parties is not changed, this inverter is enough to extract all information for the attacking party. In World_1, instead of having an inverter $Inv_A = ((f_1^A)^{-1}, (f_2^A)^{-1})$, there is only an inverter $(f_1^A)^{-1}$ to break the receiver's privacy. Similarly, in World_2 we remove Inv_B and instead, there is an inverter $(f_2^B)^{-1}$ to break the sender's privacy. The input protocols are used only in one direction, i.e., the sender of the combined **OT** acts as sender in all implementations. The sender in World_1 will never need an inverter $(f_2^A)^{-1}$ to break the sender's privacy, since he acted as sender and always chose b_0, b_1 himself. Therefore, the information for the sender is the same in the BareWorld and in World_1 and therefore, any sender attack in the BareWorld leads to an attack in World_1. Similarly, the receiver in World_2 will never need $(f_1^B)^{-1}$ to break the receiver's privacy, because he acted as receiver and knows c himself. Therefore, the information for the receiver is the same in BareWorld and in World_2. Thus, any receiver attack in the BareWorld leads to an attack in World_2. □

The impossibility proof is for combiners handling $\{(\mathtt{ot}^{sr}, \mathtt{ot}^{s}), (\mathtt{ot}^{r}, \mathtt{ot}^{sr})\}$ only. The uniform combiner even has to accept input implementations from $\mathcal{P}(\mathtt{ot}^{sr}, \mathtt{ot}^{s}) \cup \mathcal{P}(\mathtt{ot}^{r}, \mathtt{ot}^{sr}) = \{(\mathtt{ot}^{sr}, \mathtt{ot}^{s}), (\mathtt{ot}^{sr}, \mathtt{ot}^{r}), (\mathtt{ot}^{s}, \mathtt{ot}^{sr}), (\mathtt{ot}^{r}, \mathtt{ot}^{sr})\}$.

Corollary 7. *There is no uni-directional transparent black-box* $\{3, 2\}$*-robust uniform combiner for oblivious transfer.*

This implies that there is no transparent black-box construction without changing the roles of the sender and the receiver in at least one instance. Therefore, in the construction of a transparent black-box $\{3, 2\}$-robust uniform combiner, using a "swap" operation [MPW06] is necessary.

A proof similar to Theorem 3 about straightforward combiners for commitment schemes works for oblivous transfer as well.

Definition 20. *A* straightforward combiner for oblivious transfer *is a transparent black-box combiner with the following properties: Alice and Bob prepare their inputs to a sequence of* **OT***-protocols and execute these sub-protocols. Then, Bob computes his output bit from the output bits of the sub-protocols.*

Theorem 15. *There is no straightforward* $(1; 2)$*-robust combiner for oblivious transfer if one candidate scheme is secure for the receiver, the other candidate scheme is secure for the sender, and at least one scheme is secure for both parties.*

6 Conclusion

For commitment schemes, we have shown that transparent black-box $(1;2)$-robust combiners cannot exist but that black-box $(1;2)$-robust combiners do exist. Also, combiners have been constructed for which information theoretic properties are guaranteed both for secure and insecure implementations. Furthermore, it has been shown that third party black-box combiners do exist if the majority of the input implementations are secure. Considering the existence of combiners for commitment schemes, the problem has been solved in a general way; however, it is unclear yet whether universal combiners for bit commitment do exist. A construction similar to the one for **OT** in [MPW06] seems difficult, because **BC** is not known to be symmetric. Furthermore, it remains an open problem to construct a more efficient black-box combiner.

Combiners for interactive proof systems seem difficult to construct. Some basic properties of combiners for proof systems have been presented in this thesis. We now know that a possible transparent black-box combiner will not be deterministic, and that any function that splits input x into multiple instances $x_1 \dots x_n$ must not rely on the witness w. However, the question is far from being solved and further investigation is necessary.

For oblivious transfer, we have shown that a "swap" operation is necessary to obtain a transparent black-box $\{3,2\}$-robust uniform combiner.

References

[AB81] C. Asmuth and G. Blakley. An efficient algorithm for constructing a cryptosystem which is harder to break than two other cryptosystems. In *Comp. and Maths. with Appls.*, volume 7, pages 447–450, 1981.

[BCC88] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[Dam98] I. Damgård. Commitment schemes and zero-knowledge protocols. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark*, pages 63–86, 1998.

[DK05] Y. Dodis and J. Katz. Chosen-ciphertext security of multiple encryption. In *Theory of Cryptography, Second Theory of Cryptography Conference*, pages 188–209, 2005.

[EG85] S. Even and O. Goldreich. On the power of cascade ciphers. *ACM Trans. Comput. Syst.*, 3(2):108–116, 1985.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[GJ90] M. Garey and D. Johnson. *Computers and Intractability; A Guide to the Theory of* **NP***-Completeness*. 1990.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *Seventeenth Annual ACM Symposium on Theory of Computing*, pages 291–304, 1985.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in **NP** have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[Gol00] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000.

[Her05] A. Herzberg. On tolerant cryptographic constructions. In *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference*, pages 172–190, 2005.

[HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HKN+05] D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On robust combiners for oblivious transfer and other primitives. In *Advances in Cryptology - EUROCRYPT 2005*, pages 96–113, 2005.

[IL89] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235, 1989.

[MP06] R. Meier and B. Przydatek. On robust combiners for private information retrieval and other primitives. In *Advances in Cryptology - CRYPTO '06*, pages 555–569, 2006.

[MPW06] R. Meier, B. Przydatek, and J. Wullschleger. Robuster combiners for oblivious transfer, 2006. submitted.

[Nao91] M. Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[Rab81] M. Rabin. How to exchange secrets with oblivious transfer. Technical Report 81, Aiken Computation Lab, Harvard University, 1981. http://eprint.iacr.org/2005/187.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sie85] T. Siegenthaler. Design of combiners to prevent divide and conquer attacks. In *Advances in Cryptology - CRYPTO '85*, pages 273–279, 1985.

[WW04] S. Wolf and J. Wullschleger. Oblivious transfer is symmetric. Cryptology ePrint Archive, Report 2004/336, 2004. http://eprint.iacr.org/2004/336.