3-round Weak Zero-knowledge Proofs for $\mathbb{NP}$

by

Dah-Yoh Lim 林達佑

B.S., Computer Science and Information Engineering (2001)
B.A., Finance (2001)
National Taiwan University

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

April 2004 [June 2004]

© Dah-Yoh Lim 林達佑, MMIV. All rights reserved.

The author hereby grants to MIT permission to reproduce and
distribute publicly paper and electronic copies of this thesis document
in whole or in part.

Author

Department of Electrical Engineering and Computer Science
April 16, 2004

Certified by

Shafi Goldwasser
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by

Arthur C. Smith
Chairman, Department Committee on Graduate Students
3-round Weak Zero-knowledge Proofs for $\mathcal{NP}$
by
Dah-Yoh Lim 林達佑

Submitted to the Department of Electrical Engineering and Computer Science on April 16, 2004, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering

Abstract

We consider an implementation of a weakened notion of zero-knowledge (weak ZK) where the simulator is also allowed to depend on the distinguisher as well, so this type of ZK entails exhibiting an efficient simulator for every efficient (Verifier, Distinguisher) pair. This notion is interesting because in many applications of ZK protocols, weak ZK is actually enough. In addition, Goldreich and Krawczyk's proof (SIAM J. Comput. 1996) of the non-existence \(^1\) of 3-round black-box ZK protocols carries over to weak ZK directly, so we know that 3-round black-box weak ZK protocols do not exist.

In this thesis we are concerned with 3-round proofs for $\mathcal{NP}$: under the standard computational Diffie-Hellman assumption, we construct a 3-round weak ZK proof for $\mathcal{NP}$ with inverse-polynomial soundness error. To the best of our knowledge, there have been two constructive results, of Hada and Tanaka (Crypto 1998) and Lepinski (MIT Master's thesis 2001) respectively, stating that assuming some non-standard assumptions, 3-round (traditional) ZK protocols (arguments or proofs respectively) for $\mathcal{NP}$ with negligible soundness error do exist.

We use the idea of intertwining Oblivious Transfer with a ZK protocol given by Lepinski to prove our result. For every verifier and distinguisher, we construct a different simulator. The technique of simulation is novel and we believe it will have future uses. For instance, our protocol is actually WI with negligible soundness error, by virtue of Feige and Shamir's result (STOC 1990) that WI protocols do compose in parallel.

Furthermore, since the first two rounds of our protocol are actually independent of the theorem to be proven, we can think of these two rounds as an interactive setup phase after which the prover can non-interactively prove theorems to the verifier.

Thesis Supervisor: Shafi Goldwasser
Title: Professor of Electrical Engineering and Computer Science

\(^{1}\) by existence we always mean the existence of a ZK protocol for some nontrivial language, a language outside of $\mathbf{BPP}$
Acknowledgements

I would like to thank Professor Shafi Goldwasser for her encouragement, endless patience and valuable advice. I am also indebted to Professor Silvio Micali for many fruitful discussions that we’ve had.

I would like to thank Matthew Lepinski for many stimulating discussions and for listening to the presentations of the results as well as suggesting improvements to it. I’m also grateful to Dr. Alon Rosen for his suggestions about the presentation of the thesis and for suggesting we look at witness-independence, and Chun-Yuan Hsiao for helpful discussions.

I am grateful for the support of the MIT AI Lab (now the MIT CSAIL together with former LCS) for my first year. Specifically, I am grateful to Dr. Howard Shrobe for supporting me the first semester and NTT research for supporting me with an NTT fellowship the second semester (any opinions, findings, conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of NTT). Also, Dr. Thomas Knight and Professors Arthur Smith, Ronald Rivest, and Madhu Sudan were extremely supportive and helped facilitate my change of research area from AI to cryptography. Of course, all these could not have been done without the guidance of my academic advisor, Professor Randall Davis. Thank you very much!
## Contents

1 Introduction 9
   1.1 Weak ZK ........................................... 11
   1.2 Previous Results on 3-round ZK Protocols ................. 12
   1.3 Our Main Result .................................... 13

2 Preliminaries 17
   2.1 Notations and Definitions ............................ 17
   2.2 The Assumption and Theorem .......................... 20

3 The 3-round Weak ZK proof for $NP$ 23
   3.1 The Protocol ........................................ 25
   3.2 The Importance of OT: Informal Discussion ............... 27
   3.3 Proof of Soundness .................................. 31
   3.4 Proof of Weak ZKness ................................ 32

4 Conclusions and Open Problems 45
   4.1 The Soundness Error ................................. 45
   4.2 The Non-Interactive Model ............................ 46
   4.3 Weak ZK and Witness-Independence ...................... 46
   4.4 Relationships of ZK to Other Notions of Security ........ 48

A The Goldreich-Levin Theorem [GoLe89] 51

B The Assumptions of [HaTa98] 53
Chapter 1

Introduction

The existence of Zero-Knowledge (ZK) protocols (arguments and proofs) in which a party, the Prover ($P$) can convince another party, the Verifier ($V$) of the validity of a statement without giving away any other information is by itself amazing.

Informally an interactive protocol (proof or argument) consisting of a Prover $P$ and a Verifier $V$ is said to be ZK (as originally defined by Goldwasser, Micali and Rackoff [GMR89]) if, in addition to the usual completeness and soundness, it also satisfies ZKness, i.e. $V$ gets nothing useful beyond the validity of the theorem. This is proven using the simulation paradigm: that the distribution of the transcripts that $V$ sees can actually be generated by a probabilistic polynomial time (PPT) machine $S$ (for Subconscious or Simulator). Thus whatever the verifier $V$ could have (efficiently) computed after seeing the transcript, it could have computed itself by simply running the (efficient) simulator $S$, and so it has gained nothing from talking to the prover $P$.

Various characteristics of ZK protocols have been studied.

1. If soundness holds against efficient provers (arbitrary unbounded provers respectively), the protocol is called an argument (or proof respectively).

2. Depending on the closeness of the simulated distribution that the efficient simulator $S$ generates to the true distribution that $P$ generates (identical, statistically close, or computationally indistinguishable), the protocol is said to achieve Perfect ZK (PZK), Statistical ZK (SZK), or Computational ZK (CZK or
just ZK), respectively.

3. Depending on whether the efficient simulator $S$ is given only black-box access to $\mathcal{V}$, the code of $\mathcal{V}$, or may depend arbitrarily on $\mathcal{V}$, the protocol is said to achieve black-box ZK (BBZK), code-power ZK (CPZK), or (traditional) ZK (as originally defined in [GMR89]), respectively.

Due to the wide applicability of ZK protocols in principle, efficiency improvements are very important. The main efficiency measure in a ZK protocol is the number of rounds of interactions that are necessary. On the one hand, the impossibility result of Goldreich and Krawczyk [GoKr90] rules out the existence of 3-round BBZK protocols; on the other hand, to the best of our knowledge, there have been two constructive results (of Hada and Tanaka [HaTa98] and Lepinski [Le01] respectively), stating that assuming some non-standard assumptions, 3-round ZK protocols (arguments or proofs respectively) for $\mathcal{NP}$ with negligible soundness error do exist.

In this thesis we consider a reasonable weakening of zero-knowledge (weak ZK), for which we give a 3-round weak ZK proof for $\mathcal{NP}$ with inverse-polynomial soundness error under the standard computational Diffie-Hellman assumption. No non-standard assumptions are required.

**Organization of the Thesis**

Immediately following this section we will look at the notion of weak ZK and then review some previous results on 3-round ZK protocols, noting that there have only been two known ways in constructing simulators. After that we move on to state our main result and briefly describe some characteristics of our 3-round weak ZK proof for $\mathcal{NP}$ with inverse-polynomial soundness error and its consequences. This includes viewing our 3-round protocol as a non-interactive one but with an interactive 2-round setup phase, and looking at WI instead of ZK, for which our protocol would give negligible soundness error by virtue of the theorem that WI protocols do compose in parallel [FeSh90].
In chapter 2 we deal with the preliminaries. We review some standard notations, and present the formal definition of weak ZK. Then we state the Diffie-Hellman assumption and our main theorem.

Chapter 3 contains the construction and the proof of the main result. First we present Blum's ZK proof for HAM (and thus for $NP$) with soundness error of $1/2$ [Bl86]. Then we present our protocol, which in a sense is a parallelization of Blum’s protocol. Following that we will describe the role that OT plays, informally proving our main result and laying down the proof structure for the later two sections, where we formally prove soundness and weak ZKness.

Chapter 4 concludes the thesis by looking at the non-interactive model and WI.

In appendix A we present the Goldreich-Levin theorem [GoLe89], which is taken from Bellare’s exposition [Be99] of it.

In appendix B we state the non-standard assumptions of [HaTa98].

1.1 Weak ZK

We consider a weaker notion of ZK (weak ZK), which was one of the weakenings of ZK considered by Dwork, Naor, Reingold and Stockmeyer [DNR+99], where the simulator is allowed to depend on the distinguisher as well. This type of ZK entails exhibiting an efficient simulator for every efficient (Verifier, Distinguisher) pair. This notion is interesting because in many applications of ZK protocols, weak ZK is actually enough. In addition, the proof of [GoKr90] of the non-existence of 3-round black-box ZK protocols carries over to weak ZK directly, so we know that 3-round black-box weak ZK protocols do not exist.

Let’s elaborate on the notion of weak ZK. Why is it enough for actual applications of ZK protocols? This is because often, the ZK protocols are designed to fool specific verifier/distinguishers and also the theorem to be proven is not a common input. For instance, in the e-cash scheme of Liskov [Li01], the customer is supposed to prove to the bank that the coins are correctly minted before the bank signs the wallet of coins. Therefore the theorem to be proven is not a common input to both parties.
but something that must be generated by the customer (prover). To protect the customer’s identity (so that anonymity is preserved), this proof of the customer to the bank must be done in a way that doesn’t reveal any useful information about the customer’s identity to the bank. If we use an interactive protocol such that for all efficient (Verifier, Distinguisher) pairs, there exists an efficient simulator that can fool the distinguisher, then this guarantees enough security, because here, the bank is both the verifier and the distinguisher. The bank couldn’t have learnt anything it couldn’t efficiently compute by itself after the interaction because it could have simply ran the efficient simulator instead, getting transcripts that it couldn’t distinguish from real transcripts. Now it is true that as the bank’s computing power grows, the original simulator used to show that the bank hasn’t gained anything might not work anymore, but that’s alright: we know that there exists another efficient simulator that can successfully fool the bank (since the original protocol was weak ZK).

1.2 Previous Results on 3-round ZK Protocols

[HaTa98] presented a non-black-box simulation technique based mainly on two non-standard assumptions (see Appendix B) of Damgård [Da91], which allowed them to construct a 3-round ZK argument for any \( \mathcal{NP} \) language with negligible soundness error. Their main assumption has a very strong flavor of reverse engineering. One of their main assumptions is (called SDHA-1): if a probabilistic polynomial-sized circuit (PPS circuit) \( C \) can on input \( p, g, g' \) (an instance of the discrete log problem), output some \( X \) and \( X' \), then there exists a non-black-box extractor that can extract from \( C \) the discrete log of \( X \) base \( g \). In other words, having such an I/O behavior testifies to the fact that it “knows” the discrete log of \( X \). Their other main assumption, SDHA-2, is of the same form but it is more complex and actually implies the first one.  

1 In [Le01], another non-black-box simulation technique was presented, allowing the construction of a 3-round ZK proof for any \( \mathcal{NP} \) language with negligible soundness error.  

\(^1\) and which is actually false [BePa04]; see Appendix B
ness error. The main assumption is that using a "suitably chosen" hash function, secure 2-round Witness Independent proofs of knowledge can be implemented. This guarantees that the simulator can extract the implicit challenge bit from the verifier and thus simulate properly. His idea relies heavily on the so called Fiat-Shamir paradigm [FiSh89] of transforming a secure ID scheme into an extremely efficient signature scheme \(^2\) which is secure if random oracles exist. This assumption is incomparable to that of [HaTa98].

In fact, there have been a number of results which show that designing protocols in the random-oracle world and then replacing the random-oracle by a "suitably chosen" hash function will not always do [CGH98, GoTa03, BBP03]. In particular, it was shown by Goldwasser and Taumann [GoTa03] that there exists ID schemes that are secure but applying the Fiat-Shamir paradigm to it results in a completely insecure signature scheme, i.e. any hash function used to remove the interaction would result in an insecure scheme.

### 1.3 Our Main Result

To the best of our knowledge, to date there have only been three ways to construct simulators: one way is through rewinding (in which the simulator tries to guess the verifier’s challenge, iterating until it does so correctly); the second is through using the verifier’s code in an indirect way (for instance in [Ba01], where the simulator commits to the verifier’s code); the last way is through some non-standard assumptions of knowledge (like those of [HaTa98] and [Le01] that we just saw, where the verifier’s only message to the prover is assumed to be a secure proof of knowledge upon which its challenge to the prover is implicitly defined and fixed).

In this thesis, we give a new simulation method, which is made possible because we are working under the weak ZK definition, i.e. the simulator can depend arbitrarily on the distinguisher as well, and it only has to output transcripts that fool this (Verifier,
Distinguisher) pair. At the core of our protocol is Blum's 3-round ZK proof for \( \mathcal{NP} \) with soundness error 1/2 [Bl86]. Our protocol in essence is like running \( c \log k \) copies of it in parallel; \(^3\) in other words we get that under the standard (computational) Diffie-Hellman assumption, our protocol is a 3-round weak ZK proof for \( \mathcal{NP} \) with inverse-polynomial soundness error. The simulator uses both the verifier and the distinguisher to determine their challenge, and thus simulate accordingly.

In more detail, we use the intertwining of the OT channels (based on the OT of Bellare and Micali [BeMi89]) with ZK protocols first given in [Le01]. The information-theoretic security of the OT against the prover guarantees inverse-polynomial soundness error; the computational security of the OT against the verifier and distinguisher helps guarantee weak ZKness. This allows the prover to send as the third message both of the answers to challenges 0 and 1 to the verifier in different OT channels, together with some form of commitment (which was the first message in Blum's original protocol). All these, of course, are relative to the theorem to be proven (the common input). But other than this third message, the first two messages in the protocol are actually independent of the common input. The first two rounds are devoted to only setting up the OT.

Therefore this use of the OT allows us to view our 3-round protocol in a different way. The first two rounds can be thought of as an interactive preprocessing stage between the prover and the verifier. After that, whenever the prover wishes to prove some theorem to the verifier, it can do so non-interactively. It turns out that after an initial 2-round interactive phase in which the prover and the verifier sets up \( c \log k \) OT channels, the prover can then prove an unbounded number of theorems to the verifier in weak ZK, each with \( 1/k^c \) soundness error.

Note that since we are in effect running \( c \log k \) copies of Blum's protocol in parallel, the soundness error that we get is not negligible but only inverse-polynomial, \( 1/k^c \), but if we are interested in WI instead, then since WI protocols do compose in parallel [FeSh90], we can get negligible soundness error. Considering our protocol in

\(^3\) as usual, \( c \) denotes and arbitrary constant and \( k \) the security parameter which in our case will be the number of vertices of the common input, a graph \( G \)
the non-interactive model just described, we can get non-interactive WI proofs for $\mathcal{NP}$. All these will be elaborated in chapter 4.
Chapter 2

Preliminaries

2.1 Notations and Definitions

We borrow most of the notation from Goldreich [Go01]. In addition we follow the notation from Bellare’s exposition [Be99] of the Goldreich-Levin result [GoLe89], which is reproduced in Appendix A. Let \( \langle a, b \rangle \) denote the inner product of \( a \) and \( b \) modulo 2, where \( |a| = |b| \). If \( n \) is any integer, let \( [n] = \{1, ..., n\} \). If \( r = (r_1, ..., r_m) \) is a sequence of \( m \) \( k \)-bit strings and \( S \subseteq [m] \) then let \( r[S] = \sum_{j \in S} r_j \), where the summation is performed component-wise mod two, so the result is a \( k \)-bit string. Let \( S_1, ..., S_{2^m} \) be a listing of all subsets of \( [m] \) in some canonical order, \( e_n \) be the vector of all zeros except at position \( n \), and let \( y^{(n)} \) denote the \( n \)-th bit of \( y \).

Below we review the definition of a machine having a predicting advantage, interactive protocols and zero-knowledge, and formally define the notion of “weak” ZK as well. In this thesis efficient computation devices will be probabilistic polynomial-sized circuits (PPS circuit), including the verifier. This formulation using circuits is probably a bit unsatisfactory, as per Goldreich’s comment [Go01], page 215. However, this is only for clarity of presentation, and we could replace all the “PPS circuit”s that occurs in definitions and theorems by the usual “PPT machines with auxiliary input”, and everything still carries through.

Definition 1 (Predicting advantage of \( \epsilon(k) \)).
Let \( \{D_1\}_k \) and \( \{D_2\}_k \) be two probability ensembles. Let \( \text{Pred} \) be a PPS circuit circuit that takes as input inputs \( \leftarrow_R \{D_1\}_k \) and outputs a bit. \( \text{Pred} \) is said to have a predicting advantage of \( \epsilon(k) \) for the bit \( b \leftarrow_R \{D_2\}_k \) iff there exists a constant \( K \) such that for all \( k > K \):

\[
Pr_{\text{inputs},b}[\text{Pred}(\text{inputs}) = b] \geq \frac{1}{2} + \frac{\epsilon(k)}{2}.
\]

**Definition 2 (Interactive Proof System).**

A pair of interactive machines \( (\mathcal{P}, \mathcal{V}) \) is called an interactive proof system for a language \( L \) iff \( \mathcal{V} \) is a PPS circuit and the following two conditions hold:

1. **Completeness:** For all \( x \in L \),

\[
Pr[(\mathcal{P}, \mathcal{V})(x) = 1] = 1.
\]

2. **Soundness:** For all \( \mathcal{P}' \), for all polynomials \( \text{poly}(\cdot) \) and all sufficiently long strings \( x \notin L \),

\[
Pr[(\mathcal{P}', \mathcal{V})(x) = 1] < \frac{1}{\text{poly}(|x|)}.
\]

What we defined above is a **proof**, in which soundness holds against arbitrarily powerful provers. The definition for an **argument** can be easily obtained just by changing the soundness statement to “For all PPS circuit families \( \mathcal{P}', \ldots \)”, so that soundness is only guaranteed if the prover is polynomially bounded.

In the definition below, let \( (\mathcal{P}, \mathcal{V}) \) be an interactive argument/proof system for some language \( L \), and let \( \text{view}_\mathcal{P}(x) \) be the random variable describing the content of the random tape of \( \mathcal{V}' \) and the messages \( \mathcal{V}' \) receives from \( \mathcal{P} \) during an interaction on common input \( x \).

**Definition 3 (Zero-knowledge (ZK)).**

We say that \( (\mathcal{P}, \mathcal{V}) \) is zero-knowledge iff for all PPS circuit families \( \mathcal{V}' \), there exists
a PPS circuit family $S_{\nu'}$ such that the following two ensembles are computationally indistinguishable:

1. $\{\text{view}^P_{\nu'}(x)\}_{x \in L}$
2. $\{S_{\nu'}(x)\}_{x \in L}$

In the definitions below, let $(P, \nu)$ be an interactive argument-proof system for some language $L$, and let $\text{view}^P_{\nu'}(x)$ be the random variable describing the content of the random tape of $\nu'$ and the messages $\nu'$ receives from $P$ during an interaction on which $P$ proves theorem $x$.

**Definition 4 (Weak ZK).**
We say that $(P, \nu)$ is **Weak ZK** iff for all PPS circuit families $\nu'$, all PPS circuit families $D$, there exists a PPS circuit family $S_{\nu',D}$ such that the following two ensembles are computationally indistinguishable to $D$:

1. $\{\text{view}^P_{\nu'}(x)\}_{x \in L}$
2. $\{S_{\nu',D}(x)\}_{x \in L}$

**Definition 5 (Black-box Weak ZK)**
We say that $(P, \nu)$ is **Black-box Weak ZK** iff for all PPS circuit families $D$, there exists a PPS circuit family $S^D_{\nu'}$ such that for all PPS circuit families $\nu'$, the following two probability ensembles are computationally indistinguishable to $D$:

1. $\{\text{view}^P_{\nu'}(x)\}_{x \in L}$
2. $\{S^D_{\nu'}(x)\}_{x \in L}$

In this thesis, we deal solely with non-black-box notions. Black-box weak ZKness is defined only to point out an important observation of Matt Lepinski: the proof of [GoKr90] of the non-existence of 3-round black-box ZK protocols carries over to this case directly, because the proof doesn’t use the distinguisher in any way, so we know that 3-round black-box weak ZK protocols do not exist. Therefore constructing a 3-round weak ZK proof for $NP$ also shows a separation of the black-box versus non-black-box world in the context of weak ZK.
2.2 The Assumption and Theorem

Definition 6 (Parameters for Discrete Log and Diffie-Hellman).
Let \( L_{p,g,Q}(1^k) \) denote the set of triples \((p, g, Q)\) such that \( p = 2 \cdot p' + 1 \) is a random \( k\)-bit co-Sophie Germain prime (for some Sophie Germain prime \( p' \)), \( g \) is a random generator in \( \mathbb{Z}_p^* \), and \( Q = g^q \) is a random element in \( \mathbb{Z}_p^* \).

Note that restricting to co-Sophie Germain primes is only for the sake of simplicity; one can also change \( p \) to a random prime, generated with \( \phi(p) = p - 1 \) in factored form using Bach's algorithm [Ba88] (then these factors should be included in the tuple as well, so that one can check in polynomial time that these parameters are generated correctly).

All operations relating to the group elements are of course computed modulo \( p \), but we usually drop that for simplicity. Small letters usually mean the discrete logs base \( g \) of the corresponding capital letters, for instance \( Q = g^q \).

Assumption CDHA (Standard Computational Diffie-Hellman Assumption)

For every PPS circuit family \( C = \{C_k\} \), for every polynomial \( \text{poly}(\cdot) \), all sufficiently large \( k \)'s,

\[
Pr_{p,g,Q,r}[C_k(p, g, Q, g') = g^q] < \frac{1}{\text{poly}(k)},
\]

where \((p, g, Q)\) is randomly chosen from \( L_{p,g,Q}(1^k) \) and \( r \) is randomly chosen from \( \mathbb{Z}_p^* \).

Note that CDHA is stronger than assuming that the discrete log problem is hard, since if one can solve discrete logs efficiently then certainly one can solve the Diffie-Hellman problem efficiently as well. We will let DDHA denote the decisional analogue of the assumption: given \( p, g, g^a, g^b, L, R \), where \( L \) (\( R \) respectively) is randomly assigned either \( g^{ab} \) or \( g^{\text{random}} \) (\( g^{\text{random}} \) or \( g^{ab} \) respectively), it is difficult to efficiently determine which of \( L \) or \( R \) is actually \( g^{ab} \).

The main result of this thesis is the following theorem.
Theorem 1  Assuming CDHA, we have 3-round weak ZK proofs for \( \mathsf{NP} \) with inverse-polynomial soundness error.
Chapter 3

The 3-round Weak ZK proof for $NP$

In this chapter we give the main result: a 3-round weak ZK proof for $NP$ with inverse-polynomial soundness error. As discussed in section 1.1, this suffices for many applications of ZK protocols. As is “typically” done in the construction of ZK protocols, we start from Blum’s 3-round ZK proof for Hamiltonicity (and thus for $NP$) with soundness error of $1/2$ [Bl86], but any protocol with similar properties can be used here, for instance the 3-coloring protocol of Goldreich, Micali and Wigderson [GMW91].

Let COMMIT and DECOMMIT be the algorithms of some secure implementation of a commitment scheme which is computationally hiding and perfectly binding. Such schemes exist based on the hardness of the discrete log problem, which is implied by the CDHA. Blum’s protocol goes as follows.

\[
\begin{align*}
\text{BLUMP} & \quad \text{BLUMV} \\
\text{selects random permutation } \pi; & \quad \text{rejects if decommitment not proper;}
\end{align*}
\]

\[
G', \pi(G) \\
N = COM_\pi \circ COM_{G'} \\
b = 0 \text{ or } 1 \\
N^0 = \text{DECOMMIT}(COM_\pi), \text{DECOMMIT}(COM_{G'}) \\
\text{or } N^1 = \text{DECOMMIT}(COM_H)
\]

\[
\begin{align*}
\text{b} & \leftarrow R \{0,1\} \\
\text{If } b = 0, \text{ accept if } \pi \text{ is a permutation that maps } G \text{ to } G', \\
\text{If } b = 1, \text{ accept if the revealed edges form a Hamiltonian cycle in } G'.
\end{align*}
\]

Figure 3-1: Structure of Blum’s 3-round ZK Proof for HAM
Common Input: A directed graph $G = (V, E)$, where $|V| = k$.

First round (BLUMP $\rightarrow$ BLUMV): BLUMP randomly chooses a permutation $\pi$ of the vertices of $G$, and sends the verifier a commitment $COM_{\pi} = \text{COMMIT}(\pi)$ and a commitment $COM_{G'} = \text{COMMIT}(\text{Edges}(\pi(G)))$ to the edges in $G' = \pi(G)$. $N = COM_{\pi} \circ COM_{G'}$ denotes the concatenation of these two commitments.

Second round (BLUMP $\leftarrow$ BLUMV): BLUMV chooses a random challenge bit $b$ and sends it to the prover.

Third round (BLUMP $\rightarrow$ BLUMV): If $b == 0$, BLUMP sends $V$ both the decommitment to $\pi$ and the decommitment to all edges in $G'$. $N^0 = \text{DECOMMIT}(COM_{\pi}) \circ \text{DECOMMIT}(COM_{G'})$ denotes the concatenation of these two decommitments. If $b == 1$, send $V$ the decommitment to every edge in $\pi(H)$, where $H$ is a Hamiltonian cycle in $G$. $N^1 = \text{DECOMMIT}(COM_H)$ denotes this decommitment.

Verification: BLUMV rejects if the decommitment(s) is/are not proper. If $b == 0$, accept if $\pi$ is a permutation that maps $G$ to $G'$, else reject. If $b == 1$, accept if the revealed edges form a Hamiltonian cycle in $G'$, else reject.

Denote the first message sent in Blum’s protocol by the prover BLUMP as $N$ and the third message sent by BLUMP as $N^0$ or $N^1$ depending on whether the challenge $b$ from the verifier in the second message was 0 or 1 respectively. From BLUMP, another algorithm BLUMPall can be defined that on input $G$, outputs the triple $(N, N^0, N^1)$. Our protocol in a sense runs Blum’s protocol $c \log k$ times independently in parallel. Subscripts $i \in [1, ... c \log k]$ will always be used to denote which parallel copy of Blum’s protocol we are talking about, so that for instance $N_i^0$ denotes the prover’s response to challenge $b_i = 0$ in the $i$-th independent copy of Blum’s protocol, where the first message sent by the prover was $N_i$.

By the black-box ZKness of Blum’s protocol, there exists a PPS circuit simulator BLUMSIM$^{(c)}$ which on input the graph $G$ and with oracle access to the verifier $V$ can
generate tuples of the form \((\text{random-tape, } M, b, M^b)\) that are indistinguishable from the view of the verifier \(V\) when interacting with the true prover \(\text{BLUMP}\).

After describing our protocol in section 3.1 immediately after this section, we will see in section 3.2 how the OT in our protocol helps us prove soundness and weak ZKness.

### 3.1 The Protocol

**Common Input:** A directed graph \(G\).

**First round** \((P \rightarrow V)\) \(P\) selects a random instance of the discrete log problem, \((p, g, Q) \leftarrow R L_{p,g,Q},\) where \(Q = g^q \mod p\), and sends it to \(V\). Note that the first prover message is independent of the common input \(G\).

**Second round** \((P \leftarrow V)\) For each \(i \in [1, \ldots, \log c k]\), \(V\) selects a random power \(\kappa_i\) in \(\mathbb{Z}_p^*\) (where \(p = 2p' + 1\)), and flips a fair coin \(b_i\). If \(b_i = 0\), set \(X_i \leftarrow g^{\kappa_i} \mod p\) and \(Y_i \leftarrow Q / g^{\kappa_i} \mod p\). Otherwise, if \(b_i = 1\), set \(Y_i \leftarrow g^{\kappa_i} \mod p\) and \(X_i \leftarrow Q / g^{\kappa_i} \mod p\).

**Third round** \((P \rightarrow V)\) For each \(i \in [1, \ldots, \log c k]\), \(P\) checks that the pairs \(X_i Y_i = Q\). \(P\) uses the pairs \((X_i, Y_i)\) and a pseudo-random generator \(F()\) (based on the hardness of discrete log, which is implied by CDHA) to set up an oblivious transfer channel [Le01] as follows. Invoke BLUMPall\((G)\) to get \((N_i, N_i^0, N_i^1)\). Select random strings \(R_{Xi}, R_{Yi}\) of length \(k\) and random powers \(\alpha_{ij}\)'s in \(\mathbb{Z}_p^*\). Let \(seed_i^X \leftarrow (R_{Xi}, X_i^{\alpha_{i1}}), \ldots, (R_{Xi}, X_i^{\alpha_{ik}})\) and \(seed_i^Y \leftarrow (R_{Yi}, Y_i^{\alpha_{i1}}), \ldots, (R_{Yi}, Y_i^{\alpha_{ik}})\) \(^1\). The first channel is \(OT_i^X \leftarrow N_i^0 \oplus F(seed_i^X)\), and the second channel is \(OT_i^Y \leftarrow N_i^1 \oplus F(seed_i^Y)\). \(P\) then sends \((N_i, OT_i^X, OT_i^Y, R_{Xi}, R_{Yi}, g^{\alpha_{i1}}, \ldots, g^{\alpha_{ik}})\) for \(i \in [1, \ldots, \log c k]\) to \(V\).

**Verification** For each \(i \in [1, \ldots, \log c k]\), on input \((N_i, OT_i^X, OT_i^Y, R_{Xi}, R_{Yi}, g^{\alpha_{i1}}, \ldots, g^{\alpha_{ik}})\), the verifier does the following. If \(b_i = 0\) (in other words \(\kappa_i\) is the discrete

\(^1\) where \((a, b)\) denotes the inner product of \(a\) and \(b\) \mod 2
\[(p, g, Q) \leftarrow R \mathbb{L}_{p,g,Q} \]

\[p, g, Q\]

\[V\]

\[G\]

\[(X_i, Y_i) \text{ for } i \in [1, ..., c \log k]\]

for each \(i \in [1, ..., c \log k]\) \(\kappa_i \leftarrow R Z_p^*; b_i \leftarrow R \{0, 1\}\); if \(b_i = 0\) let \(X_i \leftarrow g^{\kappa_i} \mod p\) and \(Y_i \leftarrow Q/g^{\kappa_i} \mod p\) if \(b_i = 1\) let \(Y_i \leftarrow g^{\kappa_i} \mod p\) and \(X_i \leftarrow Q/g^{\kappa_i} \mod p\)

if for each \(i \in [1, ..., c \log k]\), \(X_i, Y_i \equiv Q\), construct the OTs:

\((N_i, N_i^0, N_i^1) \leftarrow \text{BLUMPall}(G)\)

\(R_{X_i}, R_{Y_i} \leftarrow R \{0, 1\}^k\)

for each \(j \in [1, ..., k]\), \(\alpha_{ij} \leftarrow R Z_p^*\)

\(seed_i^X \leftarrow \langle R_{X_i}, X_i^{\alpha_{i1}} \rangle, \ldots, \langle R_{X_i}, X_i^{\alpha_{ik}} \rangle\)

\(seed_i^Y \leftarrow \langle R_{Y_i}, Y_i^{\alpha_{i1}} \rangle, \ldots, \langle R_{Y_i}, Y_i^{\alpha_{ik}} \rangle\)

\(OT_i^X \leftarrow N_i^0 \oplus F(seed_i^X)\)

\(OT_i^Y \leftarrow N_i^1 \oplus F(seed_i^Y)\)

else reject

\((N_i, OT_i^X, OT_i^Y, R_{X_i}, R_{Y_i}, g^{\alpha_{i1}}, \ldots, g^{\alpha_{ik}}) \text{ for } i \in [1, ..., c \log k]\)

for each \(i \in [1, ..., c \log k]\), uses its knowledge of one of the discrete logs of \(X_i\) or \(Y_i\) to look into one channel of each OT pair:

if \(b_i = 0\), let \(seed_i \leftarrow \langle R_{X_i}, (g^{\alpha_{i1}})^{\kappa_i} \rangle, \ldots, \langle R_{X_i}, (g^{\alpha_{ik}})^{\kappa_i} \rangle\)

\(ans_i \leftarrow OT_i^X \oplus F(seed_i)\)

if \(b_i = 1\), let \(seed_i \leftarrow \langle R_{Y_i}, (g^{\alpha_{i1}})^{\kappa_i} \rangle, \ldots, \langle R_{Y_i}, (g^{\alpha_{ik}})^{\kappa_i} \rangle\)

\(ans_i \leftarrow OT_i^Y \oplus F(seed_i)\)

verify the message in the OT:

reject if the decommitment(s) in \(ans_i\) is/are not proper wrt \(N_i\)

if \(b_i = 0\), accept for this \(i\) if \(\pi \in ans_i\) is a permutation that maps \(G\) to \(G' \in ans_i\).

if \(b_i = 1\), accept for this \(i\) if the revealed edges in \(ans_i\) form a Hamiltonian cycle in \(G'\).

if accepted for each \(i \in [1, ..., c \log k]\), accept

else reject

Figure 3-2: Structure of the Protocol
log of $X_i$, let $seed_i \leftarrow \langle R_{X_i}, (g^{a_{x_i}})^{\kappa_i} \rangle, ..., \langle R_{X_i}, (g^{a_{x_i}})^{\kappa_i} \rangle$ and $ans_i \leftarrow OT_i^{X} \oplus F(seed_i)$; else (if $b_i == 1$, i.e. $\kappa_i$ is the discrete log of $Y_i$), let $seed_i \leftarrow \langle R_{Y_i}, (g^{a_{x_i}})^{\kappa_i} \rangle, ..., \langle R_{Y_i}, (g^{a_{x_i}})^{\kappa_i} \rangle$ and $ans_i \leftarrow OT_i^{Y} \oplus F(seed_i)$. $V$ accepts iff for each $i \in [1, ..., c \log k]$, $ans_i$ is a correct response to challenge $b_i$ with respect to $N_i$, i.e. if $b_i == 0$, $V$ checks that $ans_i$ is the proper decommitment to $N_i$, thus getting a permutation $\pi$ and a graph $G'$, and checks that $\pi(G) == G'$. If $b_i == 1$, $V$ checks that $ans_i$ is the proper decommitment to a subset of edges in the commitment $N_i$, and that these edges form a cycle.

**Theorem 1** Assuming CDHA, we have 3-round weak ZK proofs for $NP$ with inverse-polynomial soundness error.

Before proving the theorem in sections 3.3 and 3.4, in the following section we first give some intuition as to the use and importance of the OT: we will discuss why it helps to guarantee soundness and weak ZKness. This will form the basis of our proof of soundness in section 3.3 and proof of weak ZKness in section 3.4.

### 3.2 The Importance of OT: Informal Discussion

The main tool in the protocol we just discussed in section 3.1 is the OT channel idea based on [BeMi89, Le01]. This allows the prover to send as the third message $(N, N^0, N^1)$ of Blum’s protocol, where $N$ is sent in the clear and $N^0, N^1$ are sent using the OT mechanism so that at most one can be read by any efficient verifier. Indeed, the first two rounds of the protocol are devoted to only setting up two OT channels, channel $X$ and channel $Y$, such that no information about which OT channel the verifier can read is given to the prover. Thus, even an unbounded prover has no clue of which channel the verifier will be able to look into. This information-theoretic security of the OT against the prover, together with the fact that if $G$ doesn’t have a Hamiltonian cycle then no prover can produce both $N^0$ and $N^1$ corresponding to an $N$, guarantees inverse-polynomial soundness error.

To achieve this, $P$ first selects and sends a random discrete log instance $p, g, g^q;$
then \( \mathcal{V} \) selects random pairs \((X_i, Y_i)\) such that it knows the discrete log either of \(X_i\) or of \(Y_i\), and sends the pairs \((X_i, Y_i)\). Note that \(\mathcal{P}\) has no information whatsoever about which discrete log \(\mathcal{V}\) knows.

Also, for weak ZKness to hold we need to be able to prove firstly that every (Verifier, Distinguisher) pair \((\mathcal{V}', \mathcal{D})\) can at most look into one channel (distinguish it from random), i.e. we need the OT to be computationally secure against the \((\mathcal{V}', \mathcal{D})\) pair. This is accomplished under the CDHA and the Goldreich-Levin result [GoLe89] (see Appendix A). Secondly, we need to show that there is an efficient way for the simulator to determine which channel \((\mathcal{V}', \mathcal{D})\) can look into, so as to be able to commit in a way that it knows how to answer in the corresponding way. This is accomplished by invoking the Goldreich-Levin result. Thirdly, we need to show that whichever channel \((\mathcal{V}', \mathcal{D})\) gets to look into, it sees something that can be efficiently generated which is indistinguishable from what it gets from the honest prover \(\mathcal{P}\). This is accomplished by noting that in our protocol, the verifier’s challenges come before the honest prover \(\mathcal{P}\) sends \((N, N^0, N^1)\). We will not use the original simulator in Blum’s protocol BLUMSIM’ but instead define a new simulator BLUMSIM that simulates transcripts of this type.

Next, we discuss these three issues one by one.

Firstly, we informally argue that every (Verifier, Distinguisher) pair \((\mathcal{V}', \mathcal{D})\) can at most look into one channel. By the ability of \((\mathcal{V}', \mathcal{D})\) to look into a channel, we mean that \((\mathcal{V}', \mathcal{D})\) can distinguish \(F(\text{seed}_i^X)\) or \(F(\text{seed}_i^Y)\) from random. If the seed to \(F()\) is random, then this means that \((\mathcal{V}', \mathcal{D})\) breaks \(F()\) and thus solves discrete logs easily, contradicting CDHA. Otherwise, the seed to \(F()\) is not random to \((\mathcal{V}', \mathcal{D})\). By a standard hybrid argument [BlMi84, GoMi84, Ya82], there exists a position \(j \in [1, ..., k]\) such that \(\langle R_{X_i}, X_{i}^{\alpha_{ij}} \rangle\) (or \(\langle R_{Y_i}, Y_{i}^{\alpha_{ij}} \rangle\)) can be predicted with non-negligible advantage. We denote this predictor by \(\text{Pred}_X^{(\mathcal{V}', \mathcal{D})}\) (or \(\text{Pred}_Y^{(\mathcal{V}', \mathcal{D})}\) respectively). By the Goldreich-Levin result, it is possible to output a poly-sized list containing \(X_{i}^{\alpha_{ij}}\) (or \(Y_{i}^{\alpha_{ij}}\) respectively). This means that there exists an efficient machine (whose algorithm will be denoted as \(\text{RECOVER}_{\text{woEQ}}^{\text{Pred}_X^{(\mathcal{V}', \mathcal{D})}}\) later) that can output \(X_{i}^{\alpha_{ij}}\) (or \(Y_{i}^{\alpha_{ij}}\) respectively, depending on which \(\text{Pred}_X^{(\mathcal{V}', \mathcal{D})}\) it gets) with non-
negligible probability.

So if \((V', D)\) can look into both channels then there exists some \(i \in [1, \ldots, c \log k]\) for which \((V', D)\) can output both \(X_i^{\alpha_i}X\) for some \(j_X \in [1, \ldots, k]\) and \(Y_i^{\alpha_i}Y\) for some \(j_Y \in [1, \ldots, k]\). Notice that in this setting where the “secret” we are trying to get from invoking the Goldreich-Levin result has a Diffie-Hellman flavor, something stronger holds: we can actually randomize over the “secret”. Given access to \((V', D)\) that can distinguish \(R_{X_i, X_i^{\alpha_i}}\) from random non-negligibly well, we can use it to distinguish \(R_{X_i, X_i^t}\) from random non-negligibly well, where \(t\) (for “target”) is a random power which is given to us implicitly, in the form of \(g^t\).

2 So with overwhelming probability, we can make \((V', D)\) distinguish some \(R_{X_i, X_i^\beta}\), where \(\beta\) is some random power. When \((V', D)\) works, since there are only a polynomial number of positions, with non-negligible probability it would be distinguishing \(R_{X_i, X_i^{\tau t}}\) from random, and using \(\text{RECOVERwoEQ}^{\text{Pred}}(V', D)\) we can recover \(X_i^{\tau t}\) and thus \(X_i^t\).

Therefore the algorithm contradicting \textbf{CDHA} (which will be called \(\text{A}_{-\text{CDHA}}\)) can be constructed easily. On input \(p, g, g^a, g^b\), pass say \(p, g, g^a\) as the first message to \((V', D)\). Then select random elements, which includes random powers \(\alpha_i\)'s. \(\text{A}_{-\text{CDHA}}\) selects a random position and inserts \(g^b\) somewhere amidst \(g^{\alpha_i}\)'s, and then invokes \(\text{RECOVERwoEQ}^{\text{Pred}}(V', D)\) to get \(X_i^b\) and \(\text{RECOVERwoEQ}^{\text{Pred}}(V', D)\) to get \(Y_i^b\), but since \(X_iY_i = g^a\), we can actually get \(X_i^bY_i^b = g^{ab}\). By the preceding arguments, \(\text{A}_{-\text{CDHA}}\) succeeds in doing so non-negligibly well, thus contradicting the \textbf{CDHA}.

Secondly, we informally argue that there is an efficient way for the simulator to determine which channel \((V', D)\) can look into, with overwhelming probability. In the Goldreich-Levin result, an equality testing oracle is used to pinpoint the right candi-

---

2 to show this, notice first that we can randomize our target by raising it to another random power \(r\): \(g^t\) is identically distributed to \(g^r\), and once we get \(X_i^{\tau t}\), we can raise it to \(r^{-1} \mod p'\) (which can be efficiently calculated from \(r\) via Euclid’s GCD algorithm) to get back \(X_i^t\). Also, notice that there must be a 1/poly fraction of third messages that \((V', D)\) works well. If we give poly² random third messages to \((V', D)\), the probability that it always fails is \((1 - \frac{1}{\text{poly}})^{\text{poly}^2} \leq e^{-\text{poly}}\), which is negligible.

3 the probability here is only non-negligible as opposed to overwhelming, corresponding to the fact that \(X_i = g^{x_i}\) and \(g^t\) are both chosen at random by other parties, so there is no way to test whether a candidate is actually \(X_i^t\) or not, unless \textbf{DDHA} is false. This means that we do not have the oracle for testing candidates in the complete Goldreich-Levin result, and therefore the probability is only non-negligible, because we can only output a randomly selected candidate in a poly-sized list; see footnotes 7 and 12 following.
date out of the poly-sized list. In the previous point, when using RECOVERwoEQ\(\text{Pred}_i^{(V',D)}\), we do not have such an oracle, since when we are trying to contradict the CDHA both random powers are unknown to us, given only implicitly as \(g^a\) and \(g^b\). Here, however, the simulator can choose the random powers \(\alpha_{ij}\) s, so we can test equality (the corresponding algorithm for recovering \(X_i^{\alpha_{ij}}\) or \(Y_i^{\alpha_{ij}}\) with the equality testing oracle EQ will be denoted \(\text{RECOVER}^{\text{Pred}_{ij}^{(V',D)},\text{EQ}}\)). Also, here we do not require that \(\text{RECOVER}^{\text{Pred}_{ij}^{(V',D)},\text{EQ}}\) actually give us \(X_i^t\) (or \(Y_i^t\) respectively), where \(t\) is some target random power. We just need it to output tuples of the form \(X_i^r\) (or \(Y_i^r\) respectively) for some \(r\) to tell us that \((V', D)\) can look into channel \(X\) (or \(Y\) respectively). So, since we can randomize over the “secret”, \(\text{RECOVER}^{\text{Pred}_{ij}^{(V',D)},\text{EQ}}\) succeeds with overwhelming probability.

Thirdly, we informally argue that whichever channel \((V', D)\) gets to look into, it sees something that can be efficiently generated that is indistinguishable from what it expects from the honest prover \(P\). By the previous two points we can assume that we know which channel (and which channel only) \((V', D)\) is going to look into. But at this point \(P\) has yet to send \(N\), the original first message in Blum’s protocol, which is a commitment. We will not use the original simulator in Blum’s protocol BLUMSIM but instead define a new simulator BLUMSIM that simulates transcripts of this type\(^4\). Consider two special “simulator”s that fools the verifier that always challenges 0 (or 1 respectively) by committing to some \(M\) such that it will be able to answer \(M^0\) (or \(M^1\) respectively). By the security of the commitment scheme, \((V', D)\) shouldn’t be able to see the difference between \((N, N^0)\) and \((M, M^0)\) (or \((N, N^1)\) and \((M, M^1)\) respectively). More precisely, \((V', D)\) shouldn’t be able to see the difference between \((rtape, p, g, Q, b, N, N^0)\) and \((rtape, p, g, Q, b, M, M^0)\) (or \((rtape, p, g, Q, b, N, N^1)\) and \((rtape, p, g, Q, b, M, M^1)\) respectively), but since \((rtape, p, g, Q, b)\) was sent before \((N, N^0, N^1)\), the former cannot dependent on the latter. Therefore BLUMSIM does not need to worry about \((rtape, p, g, Q, b)\).

Let \(\chi_k\) be the complete graph with \(k\) vertices. Below we give the code of BLUMSIM.

\(^4\) as we will see later, this will be the “base” simulator that our simulator \(S_{V', D}\) uses
\textbf{BLUMSIM} \((G,b)\)

1. if \(b == 0\) then
   
2. \(\pi \leftarrow_R \text{Sym}_k;\)
3. \(G' \leftarrow \pi(G);\)
4. \(\text{COM}_\pi \leftarrow_R \text{COMMIT}(\pi);\)
5. \(\text{COM}_{G'} \leftarrow_R \text{COMMIT}(\text{Edges}(G'));\)
6. \(M \leftarrow \text{COM}_\pi \circ \text{COM}_{G'};\)
7. \(\text{DECOM}_\pi \leftarrow \text{DECOMMIT}(\text{COM}_\pi);\)
8. \(\text{DECOM}_{G'} \leftarrow \text{DECOMMIT}(\text{COM}_{G'});\)
9. \(M^0 = \text{DECOM}_\pi \circ \text{DECOM}_{G'};\)
10. else if \(b == 1\) then

11. \(\pi \leftarrow_R \text{Sym}_k;\)
12. \(G' \leftarrow \chi_k;\)
13. \(\text{COM}_\pi \leftarrow_R \text{COMMIT}(\pi);\)
14. \(\text{COM}_{G'} \leftarrow_R \text{COMMIT}(\text{Edges}(G'));\)
15. \(M \leftarrow \text{COM}_\pi \circ \text{COM}_{G'};\)
16. \(H \leftarrow\) a random cycle in \(\chi_k;\)
17. \(\text{COM}_H \leftarrow\) the subset of commitments in \(\text{COM}_{G'}\) corresponding to \(H;\)
18. \(\text{DECOM}_H \leftarrow \text{DECOMMIT}(\text{COM}_H);\)
19. \(M^1 \leftarrow \text{DECOM}_H;\)
20. \textbf{output}( M, M^b )

In conclusion, using the OT in this form allows decreasing the soundness error to \(1/2^{c \log k} = 1/k^c\) without compromising the (weak) ZKness.

We are now ready to prove the main theorem.

### 3.3 Proof of Soundness

Assume that \(G\) does not have a \textit{HAM} cycle, and that \(V\) is honest. Soundness holds because of the following. Consider the \(i\)-th independent copy of the parallelization \(^5\). Firstly, for any \(N_i\), which is a commitment to every edge of a randomly permuted version of \(G\), any prover can only answer at most one of the challenges of the verifier.

\(^5\) recall that in a sense, our protocol is a parallelization of Blum's protocol, running \(k\) copies of it independently in parallel; the subscript \(i\) will always be used to denote the \(i\)-th copy in such a parallelization
Secondly, the messages sent to the prover are just pairs of group elements $(X_i, Y_i)$ that multiply to $Q$. This gives completely no information to the prover about which discrete log the verifier knows. These points are elaborated below.

If $N_i$ is really a commitment to every edge of a randomly permuted version of $G$, then the prover will be able to decommit to every edge and also provide the random permutation used (this is message $N_i^0$), but there is no way for it to decommit to a random $HAM$ cycle in the permuted graph (this is message $N_i^1$) since this would imply that a $HAM$ cycle exists in $G$ as well.

If $N_i$ is not a commitment to every edge of a randomly permuted version of $G$, then the prover has the freedom to select the bogus graph such that it will be able to construct $N_i^1$; but there is no way for it to construct $N_i^0$, since otherwise this would imply that $N_i$ is actually a commitment to every edge of a randomly permuted version of $G$.

Also, the messages sent to the prover are completely independent of whether $V$ knows the discrete log of $X_i$ or $Y_i$. This means that the prover has to construct the OT channels without knowing which channel $V$ is going to see. For each $V$ message, there is a probability of 1/2 (over the choice of $V$'s random tape) that it can look into the first channel. For each $i \in [1, ..., c \log k]$, it has probability 1/2 of cheating (by guessing which channel $V$ can look at and thus selecting $N_i$ so that it can generate and put $N_i^0$ or $N_i^1$ in the corresponding channel, and filling the other channel with some other garbage), and thus the overall soundness is $(1/2)^{c \log k} = 1/k^c$.

### 3.4 Proof of Weak ZKness

Assume that the prover $P$ is honest, and trying to prove a true theorem $"G \in HAM"$ to the verifier $V$.

For any cheating pairs of PPS circuit families $(V', D)$, we show that there exists some efficient simulator $S_{V', D}(G)$ which for all graphs $G \in HAM$ of sufficient length, can produce a simulated view which is computationally indistinguishable from $\text{view}_{V'}^P(G)$ with respect to $D$. 

32
We first note that if in an execution of $(P, V')$, $P$ chooses parameters $(p, g, Q) \leftarrow R \text{L}_{p, g, Q}$, and then $V'$ replies with pairs $(X_i, Y_i)$ such that $X_iY_i \neq Q$ for some $i \in [1, ..., c\log k]$, then $P$ rejects. Therefore this type of transcript is easy to simulate, and we can assume that this doesn’t happen.

Following the structure of the informal discussions under section 3.2, firstly, we show that under the CDHA and using the Goldreich-Levin result, every (Verifier, Distinguisher) pair $(V', D)$ can at most look into one channel. Secondly, we show that using the Goldreich-Levin result, we can construct an extractor machine $D_E$ that determines which channel $(V', D)$ can look into except with negligible probability. Then we will define a simulator $S_{V', D}$ that uses $D_E$. Thirdly, we show that under CDHA and using the fact that in our protocol, the verifier’s challenges come before the honest prover $P$ sends $(N, N^0, N^1)$, whichever channel $(V', D)$ gets to look into, it sees something that can be efficiently generated by $S_{V', D}$ which is indistinguishable (to $D$) from the true transcript, otherwise some contradiction results.

First, every (Verifier, Distinguisher) pair $(V', D)$ can at most look into one channel. To simplify the argument let’s first look at only one pair of the OT, the $i$-th pair. After that we will see how the same argument can be carried forward in the case where we have $c\log k$ pairs of OT. Here, we merely need to note that although the algorithms we are going to define are trying to extract which channel $(V', D)$ can look into for a single OT, they still have to get as input $c\log k$ pairs of OT, because these algorithms run $(V', D)$, which expects $c\log k$ pairs. Because there are only a polynomial number of possibilities, we can just guess and try; the variable guess in the algorithms following takes care of this guessing. To simplify the notation, we will write $\overrightarrow{X}$ to denote $X_1, ..., X_{c\log k}$, $\overrightarrow{Y}$ to denote $Y_1, ..., Y_{c\log k}$, $g^x$ to denote $g^{a_1}, ..., g^{a_{c\log k}, k}$, $\overrightarrow{R_X}$ to denote $R_{X_1}, ..., R_{X_{c\log k}}$, and $\overrightarrow{R_Y}$ to denote $R_{Y_1}, ..., R_{Y_{c\log k}}$. In addition, since some of the algorithms need to change $R_{X_i}$ and $R_{Y_i}$ by either modifying it or adding something to it, we will write $\overrightarrow{R_{X_i}}(t)$ to denote $R_{X_1}, ..., R_{X_{i-1}}(t), R_{X_{i+1}}, ..., R_{X_{c\log k}}$, and $\overrightarrow{R_{Y_i}}(t)$ to denote $R_{Y_1}, ..., R_{Y_{i-1}}(t), R_{Y_{i+1}}, ..., R_{Y_{c\log k}}$. For instance $t$ could be $R_{X_i} + a$ (or $R_{Y_i} + a$ respectively).

Assume for the sake of contradiction that there exists an efficient pair $(V', D)$ such
that \((V', D)\) can distinguish both channels, channel \(X_i\) and channel \(Y_i\), from random for infinitely many \(ks\). Denote this set of \(ks\) as \(S_1\). This means that for each \(k \in S_1\) there exists an efficient machine \(\text{Pred}_{X}(V', D)\) that predicts the bit \(\langle R_{X_i}, X^{a_{ij|x}} \rangle\) for some \(j_X \in [1, ..., k]\) and an efficient machine \(\text{Pred}_{Y}(V', D)\) that predicts the bit \(\langle R_{Y_i}, Y^{a_{ij|y}} \rangle\) for some \(j_Y \in [1, ..., k]\) non-negligibly well. We show that this contradicts the CDHA.

In order to define the algorithm \(\text{A}_{\text{CDHA}}\) contradicting CDHA, we need to first define \(\text{RECOVERtwoEQ}^{\text{Pred}_{0}(V', D)}\), an efficient algorithm that can output \(X^{a_{ij}}\) (or \(Y^{a_{ij}}\) respectively, depending on which \(\text{Pred}_{0}(V', D)\) it gets) with non-negligible probability. This algorithm uses another algorithm \(\text{STRONG} - \text{SC}^{\text{Pred}_{0}(V', D)}\) which is taken from Bellare’s exposition [Be99] of the Goldreich-Levin result, reproduced in Appendix A. Let \(\epsilon_X\) be the advantage that \(\text{Pred}_{X}(V', D)\) can output the bit \(\langle R_{X_i}, X^{a_{ij|x}} \rangle\) for some \(i \in [1, ..., c \log k]\), \(j_X \in [1, ..., k]\), i.e. \(Pr[\text{Pred}_{X}(V', D)(G, p, g, Q, \overrightarrow{X}, \overrightarrow{Y}, aux, g^i, \overrightarrow{R_X}, \overrightarrow{R_Y}, guess) = \langle R_{X_i}, X^{a_{ij|x}} \rangle] \geq \frac{1}{2} + \frac{\epsilon_X}{2}\), where the probability is taken over random \(p, g, Q, g^i, \overrightarrow{R_X}, \overrightarrow{R_Y}\) and the random coin tosses of \((V', D)\) and \(\text{Pred}_{X}(V', D)\). Define \(\epsilon_Y\) analogously.

The algorithm \(\text{STRONG} - \text{SC}^{\text{Pred}_{0}(V', D)}\) below attempts to compute \(\langle R_{X_i}, X^{a_{ij}} \rangle\) or \(\langle R_{Y_i}, Y^{a_{ij}} \rangle\), given a random sequence of \(k\)-bit strings \(\overrightarrow{r} = (r_1, ..., r_m)\) and auxiliary bits \(b_1, ..., b_m\).\(^9\)

\[\text{STRONG} - \text{SC}^{\text{Pred}_{0}(V', D)}\]  
\[
\begin{align*}
\text{sum} &\leftarrow 0; \\
\text{for} \ i &\leftarrow 1, ..., 2^m \ \text{do} \\
& c_i \leftarrow \text{Pred}_{0}(V', D)(G, p, g, Q, \overrightarrow{X}, \overrightarrow{Y}, aux, g^i, \overrightarrow{R_X}, \overrightarrow{R_Y}, guess; r_1, ..., r_m, b_1, ..., b_m) \\
& \text{sum} \leftarrow \text{sum} + c_i; \\
\text{if} \ \text{sum} \geq 2^m/2 \ \text{then return} \ 1 \\
\text{else return} \ 0
\end{align*}
\]

\(\text{Pred}_{0}(V', D)\) takes as input \((G, p, g, Q, \overrightarrow{X}, \overrightarrow{Y}, aux, g^i, \overrightarrow{R_X}, \overrightarrow{R_Y}, guess)\) and outputs \(\langle R_{X_i}, X^{a_{ij|x}} \rangle\) (or \(\langle R_{Y_i}, Y^{a_{ij|y}} \rangle\), for some \(i \in [1, ..., c \log k]\) and \(j_X, j_Y \in [1, ..., k]\) without access to an oracle for testing the whether a candidate is \(X^{a_{ij}}\) (or \(Y^{a_{ij}}\) respectively) or not; see footnote 12 following.

\(\text{The reader might wish to review some of the notation given in section 2.1}\)

\(\text{m is a parameter set to log}(2k \min(\epsilon_X, \epsilon_Y)^{-2})\) to ensure that the probability of \(\text{STRONG} - \text{SC}^{\text{Pred}_{0}(V', D)}\) failing is at most \(1/2k\); this follows directly from lemma 1 following.

\(\text{additional inputs to the algorithm} \ \text{STRONG} - \text{SC}^{\text{Pred}_{0}(V', D)} \ \text{are specific to this protocol}\)

34
At line 4, \( \text{sum} \) is an integer counter and the "+" in "\( \text{sum} + c_i \)" is integer addition; all the other operations in the algorithm is the usual mod two operations. The following lemma is proven in [Be99].

**Lemma 1** Let \( \epsilon_X \) be the advantage that \( \text{Pred}_X^{(\gamma, \mathcal{D})} \) can output the bit \( \langle R_X, X_i^{\alpha_{ijx}} \rangle \). Then for any \( R_X, R_Y \in \{0,1\}^k \), we have

\[
P[r_{r_1}, ..., r_m \mid \text{STRONG} - \text{SC}_{\text{Pred}_X^{(\gamma, \mathcal{D})}}(G, p, g, Q, \bar{X}, \bar{Y}, aux, g^\alpha, \bar{R}_X, \bar{R}_Y, guess; r_1, ..., r_m; b_1, ..., b_m) \\
\neq \langle R_X, X_i^{\alpha_{ijx}} \rangle \leq \frac{1}{2m \epsilon_X^2},
\]

where \( r_1, ..., r_m \leftarrow_R \{0,1\}^k \). An analogous lemma holds true if we consider channel \( Y \) instead of channel \( X \).

Algorithm \( \text{RECOVERwoEQ}_{\text{Pred}_X^{(\gamma, \mathcal{D})}} \) below attempts to compute \( X_i^{\alpha_{ij}} \) (or \( Y_i^{\alpha_{ij}} \)) by calling \( \text{STRONG} - \text{SC}_{\text{Pred}_X^{(\gamma, \mathcal{D})}} \).

\[\text{RECOVERwoEQ}_{\text{Pred}_X^{(\gamma, \mathcal{D})}}(G, p, g, Q, \bar{X}, \bar{Y}, aux, g^\alpha, \bar{R}_X, \bar{R}_Y, guess)\]

1. \( r_1 \leftarrow_R \{0,1\}^k, ..., r_m \leftarrow_R \{0,1\}^k; \) candidates \( \leftarrow \{\} \);
2. for \( l \leftarrow 1, ..., 2^m \) do
3. Let \( b_1, ..., b_m \) be the binary representation of \( l - 1 \)
4. for \( n \leftarrow 1, ..., k \) do
5. \( y^{(n)} \leftarrow \text{STRONG} - \text{SC}_{\text{Pred}_X^{(\gamma, \mathcal{D})}}(G, p, g, Q, \bar{X}, \bar{Y}, aux, g^\alpha, \bar{R}_X, \bar{R}_Y, guess; r_1, ..., r_m; b_1, ..., b_m) \);
6. \( y \leftarrow y^{(1)} \circ ... \circ y^{(k)} \);
7. candidates \( \leftarrow \) candidates \( \cup \{y\} \);
8. \( l \leftarrow_R [1, ..., \| \text{candidates} \|] \);
9. return candidates(l);

**Lemma 2** Let \( \epsilon_X \) be the advantage that \( \text{Pred}_X^{(\gamma, \mathcal{D})} \) can output the bit \( \langle R_X, X_i^{\alpha_{ijx}} \rangle \), and candidates be the list of candidates obtained at the end of the for loop from line 3 to line 8 in \( \text{RECOVERwoEQ}_{\text{Pred}_X^{(\gamma, \mathcal{D})}} \). Then

\[
P[X_i^{\alpha_{ijx}} \notin \text{candidates}] \leq \frac{k}{2m \epsilon_X^2},
\]

where the probability is taken over the coin tosses of the algorithm \( \text{RECOVERwoEQ}_{\text{Pred}_X^{(\gamma, \mathcal{D})}} \).
itself.

**Proof of Lemma 2** This lemma follows directly from lemma 1 and the union bound, since the algorithm \( \text{RECOVERwoEQ}_{X}^{\text{Pred}(\mathcal{V}, \mathcal{D})} \) invokes \( \text{STRONG - SC}_{X}^{\text{Pred}(\mathcal{V}, \mathcal{D})} \) a total of \( k \) times. An analogous lemma holds when we look at channel \( Y \) instead of \( X \).

**Lemma 3** Let \( \epsilon_X \) be the advantage that \( \text{Pred}_X^{(\mathcal{V}, \mathcal{D})} \) can output the bit \( \langle R_X, X_{\alpha_{ij}X} \rangle \).

Then

\[
Pr \left[ \text{RECOVERwoEQ}_{X}^{\text{Pred}(\mathcal{V}, \mathcal{D})} (G, p, g, Q, \overrightarrow{X}, \overrightarrow{Y}, aux, g^a, \overrightarrow{R}_X, \overrightarrow{R}_Y, guess) = X^{\alpha_{ij}X} \right] \geq \frac{1}{2^m} \left( 1 - \frac{k}{2^m \epsilon_X^2} \right),
\]

where the probability is taken over the coin tosses of the algorithm \( \text{RECOVERwoEQ}_{X}^{\text{Pred}(\mathcal{V}, \mathcal{D})} \) itself.

**Proof of Lemma 3** This lemma follows directly from lemma 2 and noting that at line 9 of \( \text{RECOVERwoEQ}_{X}^{\text{Pred}(\mathcal{V}, \mathcal{D})} \), a random candidate is chosen from a list of \( 2^m \) candidates. Again, an analogous lemma holds when we look at channel \( Y \) instead of \( X \).

This algorithm is used by \( A_{\text{-CDHA}} \) to break the CDHA.

\( A_{\text{-CDHA}} (G, p, g, g^a, g^b) \)

1. \( ((X_1, Y_1), \ldots, (X_{c \log k}, Y_{c \log k}), aux) \leftarrow_R \mathcal{V} (G, rtape, p, g, g^a); \)
2. Checks that for all \( (X_i, Y_i) \) pairs multiply to \( g^a \), otherwise abort;
3. \( R_{X_1}, \ldots, R_{X_{c \log k}} \leftarrow_R \{ 0, 1 \}^k; \)
4. \( R_{Y_1}, \ldots, R_{Y_{c \log k}} \leftarrow_R \{ 0, 1 \}^k; \)
5. \( \alpha_1 \leftarrow_R \mathbb{Z}_p^*, \ldots, \alpha_{c \log k, k} \leftarrow_R \mathbb{Z}_p^*; \)
6. \( j \leftarrow_R [1, \ldots, k]; \)
7. \( g^{\alpha_{ij}} \leftarrow g^b; \)
8. \( guess \leftarrow_R 1, \ldots, c \log k \)
9. \( \text{candidate}_X \leftarrow \text{RECOVERwoEQ}_{X}^{\text{Pred}(\mathcal{V}, \mathcal{D})} (G, p, g, g^a, \overrightarrow{X}, \overrightarrow{Y}, aux, g^a, \overrightarrow{R}_X, \overrightarrow{R}_Y, guess); \)
10. \( \alpha_{11} \leftarrow_R \mathbb{Z}_p^*, \ldots, \alpha_{kk} \leftarrow_R \mathbb{Z}_p^*; \)

36
\begin{align*}
  &j \leftarrow_R [1, \ldots, k]; \\
  &g^{\alpha_{ij}} \leftarrow g^b; \\
  &\text{guess} \leftarrow_R 1, \ldots, c \log k \\
  &\text{candidate}_Y \leftarrow \text{RECOVERwoEQ}^{(\nu', \mathcal{D})}_{\text{Pred}_X}(G, p, g, g^a, X, Y, aux, g^r, R_X, R_Y, \text{guess}); \\
  &\text{candidate}_{XY} \leftarrow \text{candidate}_X \cdot \text{candidate}_Y; \\
  &\text{output} \text{candidate}_{XY}
\end{align*}

**Lemma 4** Let $\epsilon_X$ be the advantage that $\text{Pred}_X^{(\nu', \mathcal{D})}$ can output the bit $\langle R_{Xi}, X_i^{\alpha_{ij}X} \rangle$. Define $\epsilon_Y$ analogously. Recall that $c$ is the constant in our protocol, which in a sense consists of $c \log k$ parallelizations of Blum’s protocol. Then

$$
Pr_{p, g, g^a, g^b} \left[ A_{-\text{CDHA}}(G, p, g, g^a, g^b) = g^{ab} \right] \geq \left( \frac{\min(\epsilon_X, \epsilon_Y)^2}{4k^{2+c}} \right)^2,
$$

where the probability is taken over the random inputs $p, g, g^a, g^b$ and the coin tosses of the algorithm $A_{-\text{CDHA}}$ itself.

**Proof of Lemma 4** By a standard hybrid argument there exists some $j_X$ such that $\text{Pred}_X^{(\nu', \mathcal{D})}$ can predict $\langle R_{Xi}, X_i^{\alpha_{ij}X} \rangle$ non-negligibly well. If $\text{Pred}_X^{(\nu', \mathcal{D})}$ works with probability at least $\frac{1}{2} + \frac{\epsilon_X}{2}$, then by lemma 3 RECOVERwoEQ$^{(\nu', \mathcal{D})}_{\text{Pred}_X}$ succeeds in outputting $X_i^{\alpha_{ij}X}$ with probability at least $\frac{1}{2m} \left( 1 - \frac{k}{2m(\epsilon_X)^2} \right)$, where the probability is taken over the random $r_1, \ldots, r_m \leftarrow_R \{0, 1\}^k$ and the random choice of a candidate to output in the $2^m$-sized list. When we set $m$ to be at least $\log(2k(\epsilon_X)^{-2})$, this probability is at least $(\frac{\epsilon_X}{2k})(\frac{1}{2}) \geq \frac{\epsilon_X}{4k^2}$. But with $1/k$ probability over the random choice of $j \in [1, \ldots, k]$, the critical position, position $j_X$, where the predictor can predict equals the randomly selected position $j$ at which $g^{\alpha_{ij}X}$ is actually $g^b$. Also, with probability at least $1/k$ the guess will match exactly what $(\nu, \mathcal{D})$ can see. This means that with probability at least $\left( \frac{\epsilon_X}{4k^2} \right)(\frac{1}{2}) = \frac{\epsilon_X}{8k^2}$, at line 8, $\text{candidate}_X$ would be $X_i^b$. Similarly, probability at least $\frac{\epsilon_Y}{4k^2}$, at line 12, $\text{candidate}_Y$ would be $Y_i^b$. Therefore, with probability at least $\left( \frac{\min(\epsilon_X, \epsilon_Y)^2}{4k^{2+c}} \right)^2$, at line 13, $\text{candidate}_{XY}$ would be $X_i^b \cdot Y_i^b = (X_i \cdot Y_i)^b = g^{ab}$, and therefore $A_{-\text{CDHA}}$ would output $g^{ab}$ with non-negligible probability, contradicting CDHA.  

Thus \((V', D)\) can only look into at most one channel. Now we show how to generalize the same argument to the case where we have \(c \log k\) OT pairs instead of just one, i.e. in this case \((V', D)\) expects to get \(c \log k\) OT pairs instead of just one. In order for us to run \((V', D)\), we need to generate "good looking" pairs. In other words in order to extract which channel \((V', D)\) can look into for the \(i\)-th OT pair, we must somehow guess the right channels that \((V', D)\) can look into for the other \(c \log k - 1\) OT pairs. Otherwise \((V', D)\) might be able to detect us cheating and just do something weird. But this is easy to overcome since there are only \(2^{c \log k - 1} < 2^{c \log k} = k^c\) possibilities to consider. We simply enumerate through all the possibilities. At least one of them is good, in which case we extract correctly which channel \((V', D)\) can look into for the \(i\)-th OT pair. We do so for each \(i \in [1, \ldots, c \log k]\). The reason that we can do so one by one is that which channel(s) \((V', D)\) can look into is fixed after the second message (if not, we can again contradict CDHA easily). The input variable \textit{guess} in the algorithms represents exactly the current guess for the pairs that haven't been extracted. The algorithm \(A_{-CDHA}\) above doesn't need to enumerate. Instead it can just guess at random because in order to contradict the CDHA we only need non-negligible probability of success. As we will see later, the extraction algorithm \(D_E\) takes care of enumerating through all the possible guesses.

For instance, in order to extract on the first OT pair we have to send a total of \(c \log k\) OT pairs to \((V', D)\). To do so, for the second OT pair up until the last, \(c \log k\)-th OT pair, we guess which channels \((V', D)\) can look into and generate these OT's accordingly. Since there are only \(2^{c \log k - 1} < 2^{c \log k} = k^c\) possibilities to consider, we simply enumerate through all the possibilities. At least one of them is good, in which case we extract correctly which channel \((V', D)\) can look into for the first OT pair. Then we fix the first OT pair and continue doing the same for the second OT pair, this time enumerating through all the possible third to last OT pairs.

Secondly, we define an algorithm \(D_E\) that \(S_{V', D}\) uses to determine which channel (if any) \((V', D)\) can look into. We argue that \(D_E\) will succeed overwhelmingly well in determining the behavior of \((V', D)\). Then we define \(S_{V', D}\) which uses \(D_E\).

We define \(D_E\) as follows. As we argued in section 3.2, given access to \((V', D)\)
that can distinguish say \( \langle R_{X_i}, X_{i}^{\alpha_{ij}} \rangle \) from random non-negligibly well \(^{11}\), then via the Goldreich-Levin result, another efficient machine can output \( X_{i}^{\alpha_{ij}} \) with overwhelming probability. We let \( \mathcal{D}_E \) be exactly one such machine. (Again, we will only define the algorithms relative to one OT pair, but as noted above we can easily generalize the same argument to the case where we have \( c \log k \) OT pairs instead.) To define \( \mathcal{D}_E \) concretely, we start by defining auxiliary algorithms \( \text{EQ} \) and \( \text{RECOVER}^{\text{Pred}_{\mathcal{D}^{(v',D)}} \cdot \text{EQ}} \).

Fix some \( i \in [1,...,c \log k] \). Algorithm \( \text{EQ} \) tests whether the last input to it is \( X_{i}^{\alpha_{ij}} \) for some \( j \in [1,...,k] \), or \( Y_{i}^{\alpha_{ij}} \) for some \( j \in [1,...,k] \), or neither.

\[
\text{EQ}(X_i, Y_i, \alpha_{i1}, ..., \alpha_{ik}, \text{candidate})
\]

1. for \( j \leftarrow 1, ..., k \) do
2. if candidate == \( X_{i}^{\alpha_{ij}} \) then return \((0, j)\)
3. else if candidate == \( Y_{i}^{\alpha_{ij}} \) then return \((1, j)\)
4. return fail

The algorithm \( \text{RECOVER}^{\text{Pred}_{\mathcal{D}^{(v',D)}} \cdot \text{EQ}} \) below attempts to compute some \( X_{i}^{\alpha_{ij}} \) or \( Y_{i}^{\alpha_{ij}} \) by using \( \text{STRONG - SC}^{\text{Pred}_{\mathcal{D}^{(v',D)}}} \).

\[
\text{RECOVER}^{\text{Pred}_{\mathcal{D}^{(v',D)}} \cdot \text{EQ}}(G, p, g, Q, \overline{X}, \overline{Y}, aux, \alpha_{11}, ..., \alpha_{c \log k \cdot k}, \overline{R_X}, \overline{R_Y}, \text{guess})
\]

1. \( r_1 \leftarrow_R \{0, 1\}^k, ..., r_m \leftarrow_R \{0, 1\}^k; \)
2. for \( l \leftarrow 1, ..., 2^m \) do
3. Let \( b_1, ..., b_m \) be the binary representation of \( l - 1 \)
4. for \( n \leftarrow 1, ..., k \) do
5. \( y^{(n)} \leftarrow \text{STRONG - SC}^{\text{Pred}_{\mathcal{D}^{(v',D)}}}(G, p, g, Q, \overline{X}, \overline{Y}, aux, g^2, \overline{R_X}^{\alpha_n}, \overline{R_Y}^{\alpha_n}, \text{guess}; r_1, ..., r_m, b_1, ..., b_m); \)
6. \( y \leftarrow y^{(1)} \circ ... \circ y^{(k)}; \)
7. if \( \text{EQ}(X_i, Y_i, \alpha_{i1}, ..., \alpha_{ik}, y) \neq \text{fail} \) then return \( y \)
8. return 0

\(^{11}\) probability over the random tape of \((v', D)\), random \( p, g, g^a, \overline{X}, \overline{Y}, \overline{g^2}, \overline{R_X}, \overline{R_Y}; \) because there are at most four possible behaviors corresponding to which channels \((v', D)\) can look into (none, channel \( X \), channel \( Y \), or both), the asymptotic statements are made relative to these behavior.

\(^{12}\) in the Goldreich-Levin result, an equality testing oracle is used to pinpoint the right candidate out of the poly-sized list. In the previous point, when using \( \text{RECOVER}^{\text{Pred}_{\mathcal{D}^{(v',D)}} \cdot \text{EQ}} \), we do not have such an oracle, since when we are trying to contradict the \text{CDHA} both random powers are unknown to us, given only implicitly as \( g^a \) and \( g^b \). Here, however, the simulator can choose the random powers \( \alpha_{ij} \) s, so we can test equality, with the corresponding algorithm for recovering \( X_{i}^{\alpha_{ij}} \) or \( Y_{i}^{\alpha_{ij}} \) denoted as \( \text{RECOVER}^{\text{Pred}_{\mathcal{D}^{(v',D)}} \cdot \text{EQ}} \).
Lemma 5 Let $\epsilon_X$ be the advantage that $\text{Pred}_X^{(V,D)}$ can output the bit $\langle R_{X_i}, X_i^{\alpha_{ij}X} \rangle$. Then

$$\Pr \left[ \text{REC}^{(V', D), \text{EQ}} (G, p, g, Q, \overrightarrow{X}, \overrightarrow{Y}, \text{aux}, \alpha_{11}, \ldots, \alpha_{c \log k, k}, \overrightarrow{R_X}, \overrightarrow{R_Y}, \text{guess}) \neq X_i^{\alpha_{ij}X} \right] \leq \frac{k}{2^{m\epsilon_X^2}},$$

where the probability is taken over the coin tosses of the algorithm $\text{REC}^{(V', D), \text{EQ}}$ itself.

Proof of Lemma 5 This lemma follows directly from lemma 1 and the union bound, since the algorithm $\text{REC}^{(V', D), \text{EQ}}$ invokes $\text{STRONG} - \text{SC}^{(V', D)}$ a total of $k$ times. An analogous lemma holds when we look at channel $Y$ instead of $X$.

The algorithm $\mathcal{D}_E$ below attempts to compute some $X_i^{\alpha_{ij}}$ or $Y_i^{\alpha_{ij}}$ for all $i \in [1, \ldots, c \log k]$ by calling $\text{REC}^{(V', D), \text{EQ}}$ $k$ independent times for each $i$, randomizing over the powers $\alpha_{ij}$ s. It also takes care of guessing which channels $(V, D)$ can see for the as yet unextracted OT pairs.

$$\mathcal{D}_E (G, p, g, Q, \overrightarrow{X}, \overrightarrow{Y}, \text{aux}, \alpha_{11}, \ldots, \alpha_{c \log k, k}, \overrightarrow{R_X}, \overrightarrow{R_Y})$$

1. for $i \leftarrow 1, \ldots, c \log k$ do
2. \hspace{1em} $Z_i \leftarrow 0^k$;
3. \hspace{1em} for iteration $\leftarrow 1, \ldots, k$ do
4. \hspace{2em} $\beta_1, \ldots, \beta_k \leftarrow_R Z_{p'}$;
5. \hspace{2em} guess $\leftarrow 0$;
6. \hspace{2em} for enum $\leftarrow 1, \ldots, 2^{c \log k - i}$ do
7. \hspace{3em} Let $\text{guess}^{(i+1)} \circ \cdots \circ \text{guess}^{(c \log k)}$ be the $c \log k - i$ bit binary representation of enum $- 1$;
8. \hspace{3em} candidate $\leftarrow \text{REC}^{(V', D), \text{EQ}} (G, p, g, Q, \overrightarrow{X}, \overrightarrow{Y}, \text{aux}, \beta_1 \alpha_{11}, \ldots, \beta_k \alpha_{ik}, \overrightarrow{R_X}, \overrightarrow{R_Y}, \text{guess})$;
9. \hspace{3em} $(\text{guess}^{(i)}, j) \leftarrow \text{EQ} (X_i, Y_i, \beta_1 \alpha_{11}, \ldots, \beta_k \alpha_{ik}, \text{candidate})$;
10. \hspace{3em} if $\text{guess}^{(i)} == 0$ then $Z_i \leftarrow X_i^{\alpha_{ij}}$; breakfor;
11. \hspace{3em} else if $\text{guess}^{(i)} == 1$ then $Z_i \leftarrow Y_i^{\alpha_{ij}}$; breakfor;
12. return $(Z_1, \ldots, Z_{c \log k})$

Lemma 6 Let $m \geq \log (2k \min(\epsilon_X, \epsilon_Y)^{-2})$. If $\text{Pred}_o^{(V', D)}$ predicts $\langle R_{X_i}, X_i^{\alpha_{ij}X} \rangle$ with advantage $\epsilon_X$, then at the end of the $i$-th iteration of the first for loop,

$$\Pr \left[ Z_i \neq X_i^{\alpha_{ij}} \right] \leq 1/2^k,$$
where the probability is over the coin tosses of the algorithm $\text{RECOVER}^{\text{Pred}}_{\text{EQ}}(\mathcal{V}, \mathcal{D}, \mathcal{E})$ and $\mathcal{D}_E$. An analogous lemma holds when we consider channel $X$ instead of channel $Y$.

**Proof of Lemma 6** From lemma 5, when we set $m$ to be at least $\log(2k \min(\epsilon_X, \epsilon_Y)^{-2})$, $\text{RECOVER}^{\text{Pred}}_{\text{EQ}}(\mathcal{V}, \mathcal{D}, \mathcal{E})$ succeeds with probability at least $1/2$, over the coin tosses of the algorithm $\text{RECOVER}^{\text{Pred}}_{\text{EQ}}(\mathcal{V}, \mathcal{D}, \mathcal{E})$ itself. Since $\mathcal{D}_E$ runs it $k$ times independently, the success probability is overwhelming, over the coin tosses of $\mathcal{D}_E$.

Now with $\mathcal{D}_E$ defined, we describe the simulator. It begins by setting the random tape of $(\mathcal{V}, \mathcal{D})$, BLUMSIM, and the extractor $\mathcal{D}_E$ uniformly at random. The simulator then mimics the first round ($\mathcal{P}$'s turn) by choosing $(p, g, Q)$ randomly from $L_{p,g,Q}(1^k)$; in the second round $\mathcal{V}$ outputs the $c \log k$ $(X_i, Y_i)$ pairs: $((X_1, Y_1), \ldots, (X_{c \log k}, Y_{c \log k}), \text{aux}) \leftarrow_R \mathcal{V}(G, \text{rtape}, p, g, Q)$. Then the third round is set up ($\mathcal{P}_3$'s turn) by selecting some random elements: $R_{X_1}, \ldots, R_{X_{c \log k}} \leftarrow_R \{0, 1\}^k; R_{Y_1}, \ldots, R_{Y_{c \log k}} \leftarrow_R \{0, 1\}^k; \alpha_{1,1} \leftarrow_R Z_{\mathcal{V}'}; \ldots, \alpha_{c \log k, k} \leftarrow_R Z_{\mathcal{V}'}$. At this point $\mathcal{D}_E$ is run on input the previously generated elements and the common input $G$ as well: $(G, p, g, Q, \overrightarrow{X}, \overrightarrow{Y}, \text{aux}, \alpha_{1,1}, \ldots, \alpha_{c \log k, k}, R_{\overrightarrow{X}}, R_{\overrightarrow{Y}})$. Depending on the output of $\mathcal{D}_E$, the simulator behaves in one of three ways, corresponding to whether $\mathcal{D}_E$ successfully outputted some $X^{\alpha_{i,j}}_i$, or some $Y^{\alpha_{i,j}}_i$, or neither. The simulator does so by running BLUMSIM defined in section 3.2, on input $b = 0$, or $b = 1$, or not running it at all, respectively.

$\mathcal{S}_{\mathcal{V}, \mathcal{D}}(G)$

1. Sets the random tapes of $\mathcal{V}, \mathcal{D}, \mathcal{D}_E, \text{BLUMSIM}$, uniformly at random;

2. $(p, g, Q) \leftarrow_R L_{p,g,Q}(1^k); \text{ Mimics first round ($\mathcal{P}_3$'s turn)}$

3. $((X_1, Y_1), \ldots, (X_{c \log k}, Y_{c \log k}), \text{aux}) \leftarrow_R \mathcal{V}(G, \text{rtape}, p, g, Q); \text{ Mimics second round ($\mathcal{V}'$'s turn)}$

4. $R_{X_1}, \ldots, R_{X_{c \log k}} \leftarrow_R \{0, 1\}^k; R_{Y_1}, \ldots, R_{Y_{c \log k}} \leftarrow_R \{0, 1\}^k; \text{ Mimics third round ($\mathcal{P}_3$'s turn)}$

5. $\alpha_{1,1} \leftarrow_R Z_{\mathcal{V}'}; \ldots, \alpha_{c \log k, k} \leftarrow_R Z_{\mathcal{V}'}$

6. $(Z_1, \ldots, Z_{c \log k}) \leftarrow_R \mathcal{D}_E(G, p, g, Q, \overrightarrow{X}, \overrightarrow{Y}, \text{aux}, \alpha_{1,1}, \ldots, \alpha_{c \log k, k}, R_{\overrightarrow{X}}, R_{\overrightarrow{Y}})$

7. **for** $i \leftarrow 1, \ldots, c \log k$ **do**

8. **for** $j \leftarrow 1, \ldots, k$ **do**

41
\[\text{seed}_i^X \leftarrow (R_{X_i}, X_i^{\alpha_1}), \ldots, (R_{X_i}, X_i^{\alpha_k});\]

\[\text{seed}_i^Y \leftarrow (R_{X_i}, Y_i^{\alpha_1}), \ldots, (R_{X_i}, Y_i^{\alpha_k});\]

if \( Z_i = X_i^{\alpha_j} \) then

\[ (M_i, M_i^0) \leftarrow \text{BLUMSIM}(G, 0); \]

\[ \overrightarrow{OT}_i^X \leftarrow M_i^0 \oplus F(\text{seed}_i^X); \]

\[ \overrightarrow{OT}_i^Y \leftarrow F(\text{seed}_i^Y); \]

else if \( Z_i = Y_i^{\alpha_j} \) then

\[ (M_i, M_i^1) \leftarrow \text{BLUMSIM}(G, 1); \]

\[ \overrightarrow{OT}_i^X \leftarrow F(\text{seed}_i^X); \]

\[ \overrightarrow{OT}_i^Y \leftarrow M_i^1 \oplus F(\text{seed}_i^Y); \]

else

\[ \overrightarrow{OT}_i^X \leftarrow F(\text{seed}_i^X); \]

\[ \overrightarrow{OT}_i^Y \leftarrow F(\text{seed}_i^Y); \]

output \((\text{tape}, p, g, Q,\)

\[ ((X_1, Y_1), \ldots, (X_{c \log k}, Y_{c \log k})), \]

\[ (M_1, \ldots, M_{c \log k}, \overrightarrow{OT}_1^X, \ldots, \overrightarrow{OT}_1^{c \log k}, \overrightarrow{OT}_1^Y, \ldots, \overrightarrow{OT}_1^{c \log k}, \]

\[ R_{X_1}, \ldots, R_{X_{c \log k}}, R_{Y_1}, \ldots, R_{Y_{c \log k}}, g^{\alpha_1}, \ldots, g^{\alpha_{c \log k}} ) \)

Now, thirdly and lastly, we prove that \( \{S_{\mathcal{V}, \mathcal{D}}(G)\}_{G \in \mathcal{H} \mathcal{A} \mathcal{M}} \) is indistinguishable from \( \{\text{view}_{\mathcal{V}}^G(G)\}_{G \in \mathcal{H} \mathcal{A} \mathcal{M}} \) to \( \mathcal{D} \). Assume for the sake of contradiction that there exists a pair of PPS circuit families \((\mathcal{V}, \mathcal{D})\) such that \( \mathcal{D} \) can distinguish \( \{S_{\mathcal{V}, \mathcal{D}}(G)\}_{G \in \mathcal{H} \mathcal{A} \mathcal{M}} \) from \( \{\text{view}_{\mathcal{V}}^G(G)\}_{G \in \mathcal{H} \mathcal{A} \mathcal{M}} \) for an infinite number of \( ks \). Then, there exists an \( i \in [1, \ldots, c \log k] \) for which \( (\mathcal{V}, \mathcal{D}) \) can distinguish a simulated transcript \( \overrightarrow{OT}_i^X, \overrightarrow{OT}_i^Y \) from the true one. For this \( i \), \( \mathcal{D}_E \) either outputs \( X_i^{\alpha_j} \) for some \( j \in [1, \ldots, k] \), or \( Y_i^{\alpha_j} \) for some \( j \in [1, \ldots, k] \), or none of them for all \( j \in [1, \ldots, k] \).

However, as we have just saw, whenever \( (\mathcal{V}, \mathcal{D}) \) distinguishes a channel from random, \( \mathcal{D}_E \) is able to output either \( X_i^{\alpha_j} \) or \( Y_i^{\alpha_j} \) with overwhelming probability. This means that if \( \mathcal{D}_E \) fails to output either, then except with negligible probability \( (\mathcal{V}, \mathcal{D}) \) cannot distinguish any channel from random. So we are left with two possible cases, either \( \mathcal{D}_E \) outputs \( X_i^{\alpha_j} \), or it outputs \( Y_i^{\alpha_j} \).

Assume first that \( \mathcal{D}_E \) outputs \( X_i^{\alpha_j} \) for some \( j \in [1, \ldots, k] \). Let \( F_i^X \) denote \( F(\text{seed}_i^X), \)
and let $F_i^Y$ denote $F(seed_i^Y)$. Consider the following hybrids $^{13}$:

<table>
<thead>
<tr>
<th>SimOTpair$_i$</th>
<th>HybridOTpair1$_i$</th>
<th>HybridOTpair2$_i$</th>
<th>TrueOTpair$_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_i^0 \oplus F_i^X$;</td>
<td>$N_i^0 \oplus F_i^X$;</td>
<td>$N_i^0 \oplus F_i^X$;</td>
<td>$N_i^0 \oplus F_i^X$;</td>
</tr>
<tr>
<td>$F_i^Y$;</td>
<td>$F_i^Y$;</td>
<td>Random$_{[F(1^k)]}$;</td>
<td>$N_i^1 \oplus F_i^Y$;</td>
</tr>
</tbody>
</table>

$\neg$ Blum’s proof $\quad \neg$CDHA $\quad \neg$CDHA

Therefore distinguishing simulated conversations and real conversations amounts to distinguishing between these pairs (note that the left-most pair is the simulated transcript, the middle ones are the hybrid pairs, and the right-most pair is the true prover-verifier transcript); but distinguishing between a pair leads to a corresponding contradiction as shown in the figure: either $(V', D)$ contradicts the result of [Bl86] that the underlying protocol is a 3-round ZK proof for $\mathcal{NP}$ with soundness $1/2$ $^{14}$, or it looks into both channels and contradicts the Computational Diffie-Hellman assumption, because we can construct $A_{\neg{CDHA}}$ as before.

By symmetry, the same proof applies if $D_E$ outputs $Y_i^{a_{ij}}$ for some $j \in [1, ..., k]$ instead.

To summarize, under the CDHA $^{15}$ and using the OT and the Goldreich-Levin result:

1. $(V', D)$ can look into at most one of the two OT channels.

$^{13}$ the figure illustrates distinguishing a simulated transcript from a true transcript, where $D_E$ extracts $X_i^\alpha_j$ for some $i \in [1, ..., clog k], j \in [1, ..., k]$; the following illustrates the case where $D_E$ extracts $Y_i^{a_{ij}}$ instead:

<table>
<thead>
<tr>
<th>SimOTpair$_i$</th>
<th>HybridOTpair1$_i$</th>
<th>HybridOTpair2$_i$</th>
<th>TrueOTpair$_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_i^X$;</td>
<td>$F_i^X$;</td>
<td>Random$_{[F(1^k)]}$;</td>
<td>$N_i^0 \oplus F_i^X$;</td>
</tr>
<tr>
<td>$M_i^1 \oplus F_i^Y$;</td>
<td>$N_i^1 \oplus F_i^Y$;</td>
<td>$N_i^1 \oplus F_i^Y$;</td>
<td>$N_i^1 \oplus F_i^Y$;</td>
</tr>
</tbody>
</table>

$^{14}$ to be more precise, this contradicts the fact that the commitment scheme used is computationally hiding

$^{15}$ which gives us a perfectly binding and computationally hiding commitment scheme and also a pseudorandom generator
2. Whenever $(V', D)$ can look into one of the OT channels $D_B$ extracts with overwhelming probability, and so the simulator $S_{V',D}$ knows which channel $(V', D)$ is going to look at.

3. When $(V', D)$ looks into one of the OT channels, what it sees in the channel is indistinguishable from that of the true transcript.

Therefore, whenever the distinguisher has any hope of distinguishing, the simulator knows which channel it is going to look at, and can simulate accordingly.

This concludes the proof of our main theorem, theorem 1.
Chapter 4

Conclusions and Open Problems

We considered an implementation of a weakening of zero-knowledge that we believe captures many of the inherent security requirements of applications. Under the standard computational Diffie-Hellman assumption CDHA and using an intertwining of OTs [BeMi89] with ZK protocols [Le01], we implemented a 3-round weak zero-knowledge proof for $\mathcal{NP}$ with inverse-polynomial soundness error. We believe that this technique of securely parallelizing ZK protocols would be useful in similar contexts as well, for instance in proving witness-independence, which was suggested to us by Alon Rosen.

In the following sections we discuss the soundness error, the non-interactive model, the notion of weak ZK versus witness-indistinguishability, and end with some relationships of ZK with other notions of security.

4.1 The Soundness Error

Notice again that our protocol has an inverse-polynomial soundness error as opposed to the ideal negligible error. As noted in chapter 3, although the algorithms constructed are trying to extract which channel $(\mathcal{V}', \mathcal{D})$ can look into for a single OT, they still have to somehow generate $c \log k$ pairs of OT, because these algorithms run $(\mathcal{V}', \mathcal{D})$, which expects $c \log k$ pairs. In order to get negligible soundness error instead,
we need to have at least \( k \) pairs of OT \(^1\) and we also have to make sure that the extracting algorithms can tap the power of \((\mathcal{V'}, \mathcal{D})\) in distinguishing. For instance, in order to extract which channel \((\mathcal{V'}, \mathcal{D})\) can look into for the first OT, we must be able to somehow first guess \(^2\) which channels \((\mathcal{V'}, \mathcal{D})\) can look into for the other \( k - 1 \) OT pairs. Otherwise \((\mathcal{V'}, \mathcal{D})\) can detect cheating and thus purposefully fail in looking into any channel for the first OT. This need for guessing the other OTs while we’re trying to extract is exactly what is stopping us from getting negligible soundness error.

4.2 The Non-Interactive Model

In our protocol, since the first two messages are independent of the common input (the theorem to be proven), we can view this interaction as a setup phase devoted to (only) setting up the OTs, after which the prover can prove theorems in weak ZK to the (Verifier, Distinguisher). Since the prover can choose the \( \alpha_{ij} \) s, effectively this means that the OT channels can be reused (as long as the prover uses random and independent \( \alpha_{ij} \) s). So there is no need for an a priori decision of the maximum number of theorems the prover might want to prove in the future. They just have to setup \( c \log k \) OT pairs in order to achieve \( 1/k^c \) soundness error for each of the future (proofs of) theorems.

4.3 Weak ZK and Witness-Independence

We believe that our proof technique might be useful beyond the notion of weak ZK. Here we consider witness-independent protocols, as suggested to us by Alon Rosen. In particular, our protocol is actually 3-round witness-independent for the binary, polynomially bounded witness relation \( \{(G, H) \in HAM\} \) where \( G \) is a directed graph and \( H \) is a witness of its Hamiltonicity (i.e. a Hamiltonian cycle). Of course, protocols in which there is only one witness or the prover is unbounded are trivially

---

\(^1\)in each pair of the OT, the cheating prover can cheat with probability 1/2; therefore if we have \( k \) pairs, the probability that the prover can successfully cheat in all the \( k \) pairs is \( 1/2^k \)

\(^2\)mere enumeration is out because there are \( 2^{k-1} \) different possibilities, which is exponential
witness-independent, so we first limit the prover to polynomially-sized circuits with auxiliary inputs \( H_1, H_2 \) which are (different) Hamiltonian cycles of \( G \). Also, we reverse the security of the commitment scheme used (in the underlying Blum’s protocol) to a perfectly hiding and computationally binding one.

We only informally provide the intuition as to why the protocol (as modified above) is witness-independent. Firstly, since the commitment scheme is perfectly hiding, no information is leaked by the \( N_i \) (the first message in Blum’s protocol, a commitment to a random permutation and a permuted copy of \( G \) under that permutation). Secondly, note that one answer to the verifier’s challenge (of 0), denoted \( N_i^0 \) (decommitment to everything), has nothing to do with the witness. Thirdly, note that since the permutation is random and the commitment to the permutation in \( N_i \) is perfectly hiding, the distribution that one can see from \( N_i^1 \), i.e. the (decommitment to the) Hamiltonian cycle in the randomly permuted copy of \( G \), is exactly the same regardless of which witness the prover started with.

Note that in this case, our protocol can easily be modified to achieve negligible soundness error (by increasing the number of OT pairs to \( k \)). This is by virtue of the fact that witness-indistinguishable protocols do compose in parallel [FeSh90].

We can combine this with the non-interactive model discussed in the previous section, and compare with the previous results known regarding witness-indistinguishability. Let’s look at ZAPs. A ZAP is a 2-round witness-indistinguishable protocol in which the first message (from the verifier to the prover) is also independent of the statements to be proven. But in ZAPs, the verifier only uses public coins, whereas here the verifier is private coin. Also, ZAPs use Non-Interactive ZK (NIZK) proofs, which is in the common random string model, whereas here we are working in the standard model.

By derandomizing the ZAP constructions of Dwork and Naor [DeNa00], Barak, Ong, and Vadhan [BOV03] were able to get a non-interactive witness-indistinguishable proof for every \( \mathcal{NP} \) language. Their result requires two assumptions: a circuit complexity assumption for the Nisan-Wigderson-type pseudorandom generator \(^3\) and a

---

\(^3\)that \( \mathcal{E} \) has a function of nondeterministic circuit complexity \( 2^{O(n)} \)

47
cryptographic assumption that one-way functions or trapdoor permutations exist. Recall that while focusing on derandomization, Nisan and Wigderson [NiWi94] proposed a weakening of the traditional definition of pseudorandom generators [BlMi84, Ya82] which still suffices for derandomization. The benefit was that such NW-type pseudorandom generators could be constructed under weaker assumptions than the traditional ones (circuit lower bounds for exponential time, rather than the existence of one-way functions). It is interesting to note that NW-type pseudorandom generators are allowed greater running time that the adversarial circuit, in contrast to traditional generators which run in some fixed poly time against all polynomially bounded adversarial circuits. This fits the weak ZK framework as well, and it would be interesting to see what other applications weak ZK might have along a similar line of thought.

4.4 Relationships of ZK to Other Notions of Security

Recently, a line of work that takes into account the physical nature of computation (instead of just treating security as a mathematical abstraction) has emerged, notably that of Micali and Reyzin [MiRe03]. A different approach has been taken in Gassend, Clarke, van Dijk and Devadas [GCvD+02]. We believe that this type of work, coupled with the work on the random oracle methodology [CGH98, GoTa03, BBP03] and that of code obfuscation [BGI+01, Ha00], will lead to a better understanding and characterization of ZK protocols. More specifically, we believe that these work will allow us to better understand various notions of ZKness (full power as originally defined in [GMR89], code power as in [Ba01], black-box power), under more realistic models which take the physical nature of computation and thus some sense of hardware security into account. This might allow us to say define a hierarchy of assumptions linking the various notions. For instance, non-interactive (i.e. 1-round) ZK protocols exists assuming a common random string that is uniformly selected (say by a trusted third party). A series of definitions or assumptions that can gradually lead us to
approach non-interactive ZK would be interesting.
Appendix A

The Goldreich-Levin
Theorem [GoLe89]

This theorem is used to protect the bit security of the OT channel in the protocol of [Le01] and our protocol as well. The setup is as follows\(^1\); all the following are taken from Bellare’s exposition [Be99].

We are given an oracle \( B_z : \{0, 1\}^k \rightarrow \{0, 1\} \) and a real number \( \epsilon > 0 \) such that

\[
Pr_r \left[ B_z(r) = \langle r, x \rangle \right] = \frac{1}{2} + \frac{\epsilon}{2},
\]

where \( r \leftarrow_R \{0, 1\}^k \). We are also given another oracle \( EQ_z : \{0, 1\}^k \rightarrow \{0, 1\} \) which given any \( y \in \{0, 1\}^k \) returns 1 if \( y = x \) and 0 otherwise.

The algorithm STRONG − SC\(^B\)(\(z; r_1, ..., r_m; b_1, ..., b_m\)) attempts to compute \( \langle z, x \rangle \) given a random sequence of \( k \)-bit strings \( \bar{r} = (r_1, ..., r_m) \) and auxiliary bits \( b_1, ..., b_m \).

**STRONG − SC\(^B\)(z; r_1, ..., r_m; b_1, ..., b_m)**

\begin{verbatim}
    sum ← 0;
    for l ← 1, ..., 2^m do
        c_l ← B_z(z + r[S_l]) - b[S_l];
        sum ← sum + c_l;
    if sum ≥ 2^m/2 then return 1
    else return 0
\end{verbatim}

\(^1\) the reader might wish to review some of the notation given in section 2.1
The algorithm RECOVER^B_x, EQ_x attempts to compute x by using this algorithm.

RECOVER^B_x, EQ_x (1^k)
1  \( r_1 \leftarrow_R \{0, 1\}^k, \ldots, r_l \leftarrow_R \{0, 1\}^k \);
2  \textbf{for } l \leftarrow 1, \ldots, 2^m \textbf{ do}
3  \quad \text{Let } b_1, \ldots, b_m \text{ be the binary representation of } l - 1
4  \quad \textbf{for } n \leftarrow 1, \ldots, k \textbf{ do}
5  \quad \quad y^{(n)} \leftarrow \text{STRONG - SC}^B_x(e_n; r_1, \ldots, r_m, b_1, \ldots, b_m);
6  \quad \quad y \leftarrow y^{(1)} \circ \ldots \circ y^{(k)};
7  \quad \textbf{if } \text{EQ}_x(y) == 1 \textbf{ then } x' \leftarrow y;
8  \textbf{return } x';

The following lemmas and the theorem are proven in [Be99].

Lemma Let \( M = 2^m \). Then for any \( z \in \{0, 1\}^k \), we have

\[
Pr_{r_1, \ldots, r_m} \left[ \text{STRONG - SC}^B_x(z; r_1, \ldots, r_m, b_1, \ldots, b_m) \neq \langle z, x \rangle \right] \leq \frac{1}{Me^2},
\]

where \( r_1, \ldots, r_m \leftarrow_R \{0, 1\}^k \).

Lemma Let \( M = 2^m \). Then

\[
Pr \left[ \text{RECOVER}^B_x, \text{EQ}_x (1^k) \neq x \right] \leq \frac{k}{Me^2}.
\]

Theorem 2 ([GoLe89]) Let \( m \) be a parameter and \( M = 2^m \). Then there is an algorithm which makes at most \( kM \) calls to its B_x oracle, at most \( M \) calls to its EQ_x oracle, has time-complexity (execution time plus size of code) at most \( O(kM^2) \) and success probability at least \( 1 - \delta \) where \( \delta = k\epsilon^{-2}/M \).

In particular, to get success probability of at least 1/2, we would set \( M = 2k\epsilon^{-2} \). In that case \( m = \log(M) = \log(k) + 2\log(\epsilon^{-1}) + 1 \). The running time is \( O(k^3\epsilon^{-4}) \) and the number of calls to the B_x oracle is at most \( O(k^2\epsilon^{-2}) \), while the number of calls to the EQ_x oracle is at most \( O(k\epsilon^{-2}) \). This is what was used in chapter 3.
Appendix B

The Assumptions of [HaTa98]

Assumption 3 (detDHA, deterministic DHA) For every PPS circuit family \( C = \{C_k\} \), for every \( p \) of the form \( 2p' + 1 \) and \( g \) a generator in the group \( \mathbb{Z}_p \), for every polynomial \( \text{poly}(\cdot) \), all sufficiently large \( k \)'s,

\[
\Pr_{r_1, r_2} \left[ C_k(p, p', g, g^{r_1}, g^{r_2}) = g^{r_1 r_2} \right] < \frac{1}{\text{poly}(k)},
\]

where both \( r_1 \) and \( r_2 \) are chosen uniformly and independently from \( \mathbb{Z}_{p'} \).

For those who have read Hada and Tanaka’s paper, please note that we have reversed the roles of \( B \) and \( X \) in the following, to make the notation consistent with the body of the thesis.

Assumption 4 (SDHA-1, Strong Diffie-Hellman Assumption-1) Let \( C = \{C_k\} \) be a PPS circuit family which takes as input \((p, p', g, g^r)\) and outputs \((X, B)\) such that \( B = X^r \), where \( C_k \) runs on \( p \) of length \( k \). For every PPS circuit family \( C = \{C_k\} \), there exists a PPS circuit family \( C_E = \{C_{E_k}\} \) which on input \((p, p', g, g^r)\) outputs \((X', B', x)\) such that:

1. For every equal setting of the random tapes of \( C_k \) and \( C_{E_k} \), for every \( p \) of the
form $2p' + 1$ and $g$ a generator in the group $\mathbb{Z}_p$,

$$Pr_r \left[ (X, B) \leftarrow C_k(p, p', g, g^r); 
\quad (X', B', x) \leftarrow C_{E_k}(p, p', g, g^r) : 
\quad (X, B) = (X', B') \right] = 1,$$

where $r$ is randomly chosen from $\mathbb{Z}_{p'}$.

2. For every $p$ of the form $2p' + 1$ and $g$ a generator in the group $\mathbb{Z}_p$, for every polynomial poly$(\cdot)$, all sufficiently large $k$'s,

$$Pr_r \left[ (X', B', x) \leftarrow C_{E_k}(p, p', g, g^r) : 
\quad B' = (X')^r \wedge X' \neq g^x \right] < \frac{1}{\text{poly}(k)},$$

where $r$ is randomly chosen from $\mathbb{Z}_{p'}$.

**Assumption 5 (SDHA-2, Strong Diffie-Hellman Assumption-2)** Let $C = \{C_k\}$ be a PPS circuit family which takes as input $(p, p', g, g^{r_1}, g^{r_2}, g^{r_1r_2})$ and outputs $(Y, A)$ such that $A = Y^{r_2}$, where $C_k$ runs on $p$ of length $k$. For every PPS circuit family $C = \{C_k\}$, there exists a PPS circuit family $C_{E_k} = \{C_{E_k}\}$ which on input $(p, p', g, g^{r_1}, g^{r_2}, g^{r_1r_2})$ outputs $(Y', A', y)$ such that:

1. For every equal setting of the random tapes of $C_k$ and $C_{E_k}$, for every $p$ of the form $2p' + 1$ and $g$ a generator in the group $\mathbb{Z}_p$, and every $r_1 \in \mathbb{Z}_{p'}$

$$Pr_{r_2} \left[ (Y, A) \leftarrow C_k(p, p', g, g^{r_1}, g^{r_2}, g^{r_1r_2}); 
\quad (Y', A', y) \leftarrow C_{E_k}(p, p', g, g^{r_1}, g^{r_2}, g^{r_1r_2}) : 
\quad (Y, A) = (Y', A') \right] = 1,$$

where $r_2$ is chosen randomly from $\mathbb{Z}_{p'}$.

2. For every $p$ of the form $2p' + 1$ and $g$ a generator in the group $\mathbb{Z}_p$, for every polynomial poly$(\cdot)$, all sufficiently large $k$'s,

$$Pr_{r_2} \left[ (Y', A', y) \leftarrow C_{E_k}(p, p', g, g^{r_1}, g^{r_2}, g^{r_1r_2}) : 
\quad A' = (Y')^{r_2} \wedge A' \neq (g^{r_2})^y \wedge A' \neq (g^{r_1r_2})^y \right] < \frac{1}{\text{poly}(k)},$$

54
where $r_2$ is chosen randomly from $\mathbb{Z}_p$.

Hada and Tanaka proved that SDHA-1 together with detDLA, the “deterministic” version of the discrete log assumption, implies detDHA. They also remarked that SDHA-2 implies SDHA-1, and the reverse implication is unclear. SDHA-1 is used to prove that their protocol is zero-knowledge, and SDHA-2 is used to prove that their protocol is sound, unless detDLA is false.

**Theorem 3 ([HaTa98])** Assuming detDLA and SDHA-2, we have 3-round ZK arguments for \textsf{NP}.

We note again that Bellare and Palacio has shown in [BePa04] that SDHA-2 is actually false, and they patch it up by using another assumption, recovering the 3-round ZK argument for \textsf{NP}.


Bibliography


