On Homomorphic Encryption and Enhanced Trapdoor Permutations

Ron Rothblum

Under the Supervision of Professor Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science

Submitted for the degree of Master of Science
to the Scientific Council of the Weizmann Institute of Science

September 2010

Abstract

In this thesis we study two remotely related cryptographic primitives: homomorphic encryption and enhanced trapdoor permutations.

Our main result regarding homomorphic encryption shows that any *private-key* encryption scheme that is weakly homomorphic with respect to addition modulo 2, can be transformed into a *public-key* encryption scheme. The homomorphic feature referred to is a minimalistic one; that is, the length of a homomorphically generated encryption should be independent of the number of ciphertexts from which it was created. Our resulting public-key scheme is homomorphic in the following sense. If $i + 1$ repeated applications of homomorphic operations can be applied to the private-key scheme, then i repeated applications can be applied to the public-key scheme.

In an independent part of the thesis, we study (enhanced) trapdoor permutations (TDPs). We note that in many setting and applications trapdoor permutations behave unexpectedly. In particular, a TDP may become easy to invert when the inverter is given auxiliary information about the element to be inverted (e.g., the random coins that sampled the element). Enhanced TDPs were defined in order to address the latter special case, but there are settings in which they apparently do not suffice (as demonstrated by the introduction of doubly-enhanced TDPs). We study the hardness of inverting TDP in natural settings, which reflect the security concerns that arise in various applications of TDPs to the construction of complex primitives (e.g., Oblivious Transfer and NIZK). For each such setting, we define a corresponding variant of the notion of an enhanced TDP such that this variant is hard to invert in that setting. This yields a taxonomy of variants, which lie between enhanced TDPs and doubly-enhanced TDPs. We explore this taxonomy and its relation to various applications.

Acknowledgements

First and foremost, I would like to express my thanks and appreciation to my advisor, Oded Goldreich, for his encouragement and guidance in all stages of this work. In particular, I thank him for introducing me to the field of Foundations of Cryptography through his fantastic courses, and for shaping my understanding of research through many helpful discussions and constructive comments. Working with Oded has been a true pleasure and I am delighted that we will be working together in the near future.

This thesis would not have been possible without the endless support of my beloved family. Although we are usually in at least three different countries, the rare times when we are all together are the most precious to me. I am extremely grateful to Guy, for always being there for me, no matter when and where he is and for setting the bar way up there. To Alex and Modi, for their kindness and caring and for their hospitality in the past, present and most importantly, the future. And to my parents, Naomi and Uri, for more than words can ever express.

Lastly, I thank my dear Lucy for her constant encouragement, for the unwaivering faith that she has in me and for her boundless enthusiasm in all matters. Most of all I thank her for understanding me even when I do not understand myself.

Contents

Chapter 1

Introduction

In this thesis we explore two important cryptographic primitives: homomorphic encryption and (enhanced) trapdoor permutations. The thesis is divided into two main chapters, one for each primitive.

In Chapter 2 we study the notion of homomorphic encryption. A more extensive introduction to homomorphic encryption is provided in Chapter 2 but loosely speaking, homomorphic encryption refers to the ability, given encryptions $E_e(m_1), \ldots, E_e(m_k)$, to generate an encryption $E_e(m^*)$ of a related message $m^* = f(m_1, \ldots, m_k)$ for some (efficiently computable) function f. Homomorphic encryption can be defined for both private and public-key encryption. The main result presented in Chapter 2 relates the notions of private and public-key homomorphic encryption by constructing a *public-key* scheme based on any *private-key* scheme that is homomorphic w.r.t addition modulo 2 while (partially) retaining the homomorphic properties of the underlying private-key scheme.

In Chapter 3, we study (enhanced) trapdoor permutations. Informally, a trapdoor permutation is an efficiently computable permutation that is hard to invert in general but becomes easy to invert given a secret trapdoor. A more detailed exposition is presented in Chapter 3, which studies the hardness of inverting trapdoor permutations in natural settings. These settings reflect some additional information that is given to the inverting algorithm that may make inverting the permutation an easy task. For each such setting we define a corresponding variant of an enhanced trapdoor permutation that is hard to invert in that setting. We explore connections between these variants and their relation to applications such as oblivious transfer and non-interactive zero-knowledge proofs.

Chapter 2

Homomorphic Encryption: from Private-Key to Public-Key

2.1 Introduction

Homomorphic encryption is a paradigm that refers to the ability, given encryptions of some messages, to generate an encryption of a value that is related to the original messages. Specifically, this ability means that from encryptions of k messages $m_1, \ldots, m_k$ it is possible to generate an encryption of $m^* = f(m_1, \ldots, m_k)$ for some (efficiently computable) function f. Ideally, one may want the homomorphically generated encryption of m^* to be distributed identically (or statistically close) to a standard encryption of m^*. We call schemes that have this property strongly homomorphic. Indeed, some proposed encryption schemes are strongly homomorphic w.r.t some algebraic operations such as addition or multiplication (e.g. Goldwasser-Micali [GM84], El-Gamal [Gam84]).

For some applications, it seems as though strongly homomorphic encryption is an overkill. There are weaker notions of homomorphic encryption that might be easier to construct and still suffice for these applications. The very minimal requirement is that a homomorphically generated encryption decrypts correctly to the corresponding message. Alas, this weak requirement does not seem to be useful as is, because it captures schemes that we do not really consider to be homomorphic: Actually, *any* encryption scheme can be slightly modified to satisfy this weak requirement w.r.t *any* efficient operation[1]. A more meaningful notion is obtained by restricting the length of the homomorphically generated encryption. Specifically, we call an encryption scheme weakly homomorphic if homomorphically generated encryptions properly decrypt to the correct message *and* their lengths depend *only* on the security parameter and the message length (and not on the number of input ciphertexts).

[1]Consider implementing the homomorphic evaluation algorithm as the identity function. That is, given ciphertexts and a description of an operation, just output both. Then, modify the decryption algorithm to first decrypt all the ciphertexts and then apply the operation to the decrypted messages. Thus, homomorphic evaluation is delegated to the decryption algorithm that, using the decryption key, can trivially evaluate the required operation.

2.1.1 Private-Key vs. Public-Key

When presenting homomorphic encryption, we did not specify whether we consider private-key or public-key encryption schemes. Indeed, one can define strong/weak homomorphic encryption in both settings (with only minor differences). The focus of this paper is showing the connection between public-key and private-key homomorphic encryption.

The easy direction is showing that a public-key homomorphic encryption scheme can be transformed into a private-key homomorphic scheme. This transformation is quite simple and involves only a minor issue. Intuitively, it seems as though any public-key homomorphic scheme *is* a private-key homomorphic scheme. The only problem is that in the public-key setting (in contrast to the private-key one), the homomorphic evaluation algorithm is also given the encryption-key. A simple transformation that addresses this issue is to append the encryption-key to each ciphertext. The resulting private-key scheme clearly retains the homomorphic properties of the public-key scheme (this holds for both strongly and weakly homomorphic schemes).

The harder direction is showing that a private-key homomorphic encryption scheme implies a public-key one. This direction will be addressed by our main result, Theorem 2.3, which shows how to construct a public-key encryption scheme from any private-key scheme that is weakly homomorphic w.r.t addition modulo 2. The resulting public-key scheme partially retains the homomorphic properties of the private-key scheme (see Section 2.1.2).

We note that it is quite easy to transform a *strongly homomorphic* private-key scheme into a strongly homomorphic public-key one. In fact, this transformation was used by Barak [Bar10] in his exposition of the work of van Dijk et al. [vDGHV10]. For further discussion, see Section 2.1.3.

2.1.2 Homomorphic Properties of the Public-Key Scheme

So far we have described homomorphic evaluation as a one-shot process, however one can consider repeated application of the homomorphic evaluation algorithm. For *strongly* homomorphic encryption it is possible to do this because homomorphically generated values are identical (or statistically close) to real ciphertexts. For *weakly* homomorphic encryption, the homomorphically generated values can completely differ from real ciphertexts, hence it is unclear that it is possible to keep computing on such homomorphically generated data. Gentry et al. [GHV10] called a scheme that supports i such repeated applications an i-hop homomorphic encryption scheme.

The public-key scheme that we construct is homomorphic in the following sense. If the original private-key scheme is $(i+1)$-hop homomorphic w.r.t some set of operations (which must include addition modulo 2), then the public-key scheme is i-hop homomorphic w.r.t the same set of operations. That is, we lose one application of the homomorphic operation in the construction.

2.1.3 Technique

The intuition for how to move from private to public key can be seen in a more straightforward manner in the case of *strongly* homomorphic schemes. The following construction was suggested implicitly in [Bar10].

Let E and D be the respective encryption and decryption algorithm of a private-key encryption scheme. Suppose that this encryption scheme is *strongly* homomorphic w.r.t the identity function. That is, it is possible to "re-randomize"[2] ciphertexts. Such a scheme can be used to construct a public-key bit-encryption scheme[3] as follows. The (private) decryption-key is a key k of the private-key scheme and the (public) encryption-key consists of an encryption of 0 and an encryption of 1 (i.e. $E_k(0)$ and $E_k(1)$). To encrypt a bit σ just re-randomize the ciphertext corresponding to σ. To decrypt, apply the private-key decryption algorithm using k (i.e. D_k).

The security of this construction follows from the fact that after re-randomization, all information on the original ciphertext, which was re-randomized, is completely lost. Since *weakly* homomorphic encryption does not guarantee this property, this transformation does not work and we use a more complicated construction, outlined next.

We construct a public-key bit-encryption scheme based on any private-key scheme that is weakly homomorphic w.r.t addition modulo 2. Our decryption key is also a key k of the private-key scheme but the public-key is no longer a single encryption of 0 and 1, but rather a sequence of *many* encryptions of each. Specifically, the public-key consists of two lists of ciphertexts; the first is a list of ℓ encryptions of 0 and the second is a list of ℓ encryptions of 1. To encrypt a bit σ we choose a *random subset* $S \subseteq [\ell]$ that has parity σ (i.e. $|S| \equiv \sigma \bmod 2$). We use S to select ℓ ciphertexts from the public key by selecting the i-th ciphertext from the first list if $i \notin S$ (and from the second if $i \in S$). By *homomorphically adding* the selected ciphertexts modulo 2, we obtain a ciphertext that correctly decrypts to σ.

Most of this work deals with showing that the construction is indeed semantically-secure. To prove security we consider, as a mental experiment, setting both lists in the public-key to be encryptions of 0. Because the mental experiment is computationally indistinguishable from the actual scheme, proving that the original scheme is secure reduces to showing that when *both* lists consist of encryptions of 0, it is essentially impossible to find the parity of the random subset used in the homomorphic encryption process.

We prove the latter via an information-theoretic theorem that may be of independent interest: Let $X_1, \ldots, X_\ell$ and $Y_1, \ldots, Y_\ell$ be independent and identically distributed over a finite set Ω and let S be a random subset of $[\ell]$. We consider the list Z, defined as $Z_i = X_i$ for $i \notin S$ and $Z_i = Y_i$ for $i \in S$. The theorem states that it is essentially impossible to guess the parity of S based on X, Y and m bits of information on Z. That is, any such guess will be correct with probability that is bounded by (roughly) $\frac{1}{2} + 2^{\ell - m}$. The proof

[2]This means that there exists an algorithm RR such that for *any* encryption c of a bit b, the output of $RR(c)$ is distributed identically to $E_e(b)$.

[3]A bit-encryption scheme is a public-key encryption scheme that only handles single-bit messages. Such schemes suffice to construct full-fledged public-key encryption schemes (see [Gol04]).

of the information-theoretic theorem makes use of the Efron-Stein decomposition [ES81], an extension of Fourier analysis for product distributions.

We mention that our construction is secure even if we use a slightly weaker definition of homomorphic encryption. Specifically, the length of homomorphically generated encryptions can be a mildly increasing function of the number of input ciphertexts.

2.1.4 Application of our Construction to Fully-Homomorphic Encryption

Our generic transformation from private-key to public-key encryption can be used as a general methodology for constructing (weakly) homomorphic public-key encryption. One application of this methodology, which actually motivated this work, is to simplify the presentation of the [vDGHV10] fully-homomorphic encryption scheme.

A fully-homomorphic encryption scheme is an encryption scheme that is homomorphic w.r.t any (efficiently computable) function. The concept of fully-homomorphic encryption was first proposed by Rivest et al. [RAD78] in the 70's, but the first concrete proposal was only made recently in the breakthrough work of Gentry [Gen09].

Building on the work of Gentry [Gen09], van Dijk et al. [vDGHV10], proposed a simpler fully-homomorphic public-key scheme. Actually, they propose several variants of the same scheme. Barak [Bar10] noted that one of these variants is in fact fully-homomorphic in the *strong* sense, that is, homomorphically evaluated encryptions are distributed statistically close to actual encryptions. However, this variant requires a stronger assumption than the other variants that are only weakly homomorphic.

From a high-level point of view, both the weak and strong variants of the fully homomorphic scheme are constructed by first proposing a simple *private-key* homomorphic scheme that is only "somewhat" homomorphic (that is, homomorphic w.r.t some restricted functions) and then showing how to modify this scheme into a somewhat homomorphic *public-key* one. The last step uses the bootstrapping technique of [Gen09] to transform the somewhat homomorphic scheme into a fully-homomorphic one.

The aforementioned modification, from private-key to public-key, uses specific properties of the [vDGHV10] scheme. We suggest to use our transformation as an alternative, where the advantage is that our transformation is generic and does not use specific properties of their scheme. Our transformation can be applied to both the strong and weak variants of the somewhat homomorphic *private-key* scheme to obtain a correspondingly strong/weak somewhat homomorphic *public-key* scheme. Note that although the somewhat homomorphic public-key scheme produced by our transformation is slightly different from the one of [vDGHV10], the last step of bootstrapping (see [Gen09]) and reducing the (multiplicative) depth of the decryption circuit can still be applied to our construction.

An alternative, and perhaps more intuitive way to present the [vDGHV10] scheme was taken by Barak [Bar10] for the strongly homomorphic variant of [vDGHV10]. Barak focuses only on presenting the simpler fully-homomorphic *private-key* scheme, since the transformation to a public-key one is easy (as described in Section 2.1.3). Using our result, it is possible to extend Barak's approach to the weakly homomorphiv variant of

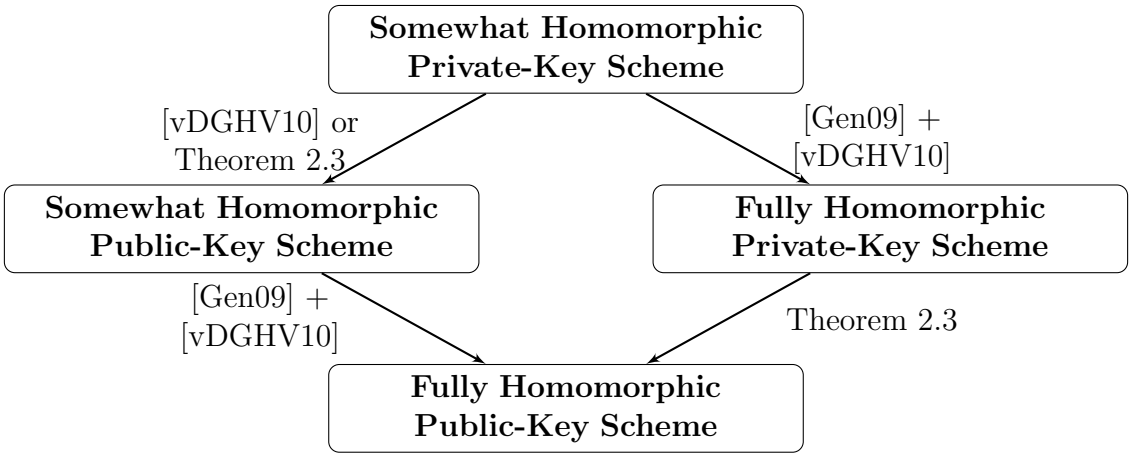

Figure 2.1: Constructing the weakly homomorphic variant of the [vDGHV10] fully-homomorphic public-key scheme.

the [vDGHV10] scheme. Thus, we suggest to simplify the presentation of the [vDGHV10] scheme by focusing only on showing a (weakly) fully-homomorphic *private-key* scheme and then, using our generic transformation, to obtain a (weak) fully-homomorphic public-key one. The two approaches to presenting the weakly homomorphic variant of the [vDGHV10] scheme, that were outlined in this section, are depicted in Figure 2.1.

2.2 Preliminaries

For a set S, we denote by $x \in_R S$ a uniformly distributed element $x \in S$. Similarly we denote by $X \subseteq_R S$ a uniformly distributed random subset of S.

Non-Standard Notation For every $\ell \in \mathbb{N}$, random variables $X = X_1, \ldots, X_\ell$ and $Y = Y_1, \ldots, Y_\ell$ and set $S \subseteq [\ell]$, we denote by $X_{\overline{S}}Y_S$, the random variable $Z = Z_1, \ldots, Z_\ell$ where $Z_i = X_i$ for $i \notin S$ and $Z_i = Y_i$ for $i \in S$.

2.2.1 Encryption Schemes

We follow notations and definitions of [Gol01, Gol04]. In particular we use their definition of semantically secure encryption schemes, both in the private-key and public-key settings. Throughout this paper we restrict our attention to bit-encryption schemes, i.e., schemes that encrypt a single bit. For simplicity, we say public-key (resp. private-key) encryption when we actually mean public-key (resp. private-key) bit-encryption.

2.2.2 Homomorphic Encryption

Since we only consider *weakly* homomorphic encryption, from here on, when we say homomorphic we always mean in the *weak* sense as defined next.

Definition 2.1. (G, E, D, H) *is a* homomorphic private-key encryption scheme *with respect to a set of families of polynomial-sized circuits* $\mathcal{C}$ *if* (G, E, D) *are a private-key encryption scheme,* H *is a probabilistic polynomial-time algorithm and there exists a polynomial* $m(\cdot)$ *such that for every circuit family* $\{C_k\}_{k\in\mathbb{N}} \in \mathcal{C}$*,* $n \in \mathbb{N}$*, polynomial* $\ell(\cdot)$*, keys* $(e, d) \leftarrow G(1^n)$*, and* $\ell = \ell(n)$ *single bit messages* $b_1, \ldots, b_\ell \in \{0, 1\}$ *the following holds:*

- *Correct decryption of homomorphically generated encryptions:*

$$D_d\left(H\left(C_\ell, E_e(b_1), \ldots, E_e(b_\ell)\right)\right) = C_\ell\left(b_1, \ldots, b_\ell\right).$$

- *The length of homomorphically generated encryptions is independent of* ℓ*:*

$$\left|H\left(C_\ell, E_e(b_1), \ldots, E_e(b_\ell)\right)\right| \leq m(n).$$

Homomorphic *public-key* encryption is defined analogously (with the modification that H gets the public encryption-key as an additional input).

2.2.3 i-Hop Homomorphic Encryption

The homomorphic evaluation algorithm in Definition 2.1 is only required to operate on ciphertexts that were output by the encryption algorithm. The definition does not specify what happens if the homomorphic evaluation algorithm is applied to its own output. Gentry et al. [GHV10] defined an i-hop homomorphic encryption scheme as a scheme for which it is possible to apply the homomorphic evaluation algorithm consecutively i times.

Let G, E, D, H be a homomorphic encryption scheme w.r.t to a set of circuit families $\mathcal{C}$. For a given encryption key e, we denote by $W_0(e)$ the set of all valid ciphertexts of the encryption scheme, i.e., all possible outputs of the encryption algorithm E_e applied to a single bit message. For $j \geq 1$, we define $W_j(e)$ to be the set of all possible outputs of the homomorphic evaluation algorithm H when applied to elements in $W_{j-1}(e)$ and a circuit $C \in \mathcal{C}$. We say that elements in $W_j(e)$ are j-th level ciphertexts.

Definition 2.2. (G, E, D, H) *is an* i-hop homomorphic private-key encryption scheme *with respect to a set of families of polynomial-sized circuits* $\mathcal{C}$ *if* (G, E, D) *are a private-key encryption scheme,* H *is a probabilistic polynomial-time algorithm and there exists a polynomial* $m(\cdot)$ *such that for every circuit family* $\{C_k\}_{k\in\mathbb{N}} \in \mathcal{C}$*,* $n \in \mathbb{N}$*, polynomial* $\ell(\cdot)$*, keys* $(e, d) \leftarrow G(1^n)$*,* $0 \leq j \leq i$*, and* $\ell = \ell(n)$*, ciphertexts* $w_1, \ldots, w_\ell \in W_j(e)$ *of level* j *the following holds:*

- *Correct decryption of homomorphically generated encryptions:*

$$D_d\left(H\left(C_\ell, w_1, \ldots, w_\ell\right)\right) = C_\ell\left(D_d(w_1), \ldots, D_d(w_\ell)\right). \tag{2.1}$$

- *The length of homomorphically generated encryptions is independent of ℓ:*

$$|H\left(C_\ell, w_1, \ldots, w_\ell\right))| \leq m(n). \tag{2.2}$$

Homomorphic *public-key* encryption is defined analogously, with the modification that H receives the encryption-key as an additional input.

2.3 Constructing a Public-Key Scheme from a Homomorphic Private-Key Scheme

In this section we show how to construct a public-key scheme based on any private-key scheme that is homomorphic w.r.t addition modulo 2.

Theorem 2.3. *Any multiple-message semantically secure private-key encryption scheme that is homomorphic with respect to addition modulo 2 can be transformed into a semantically secure public-key encryption scheme. Furthermore, if the private-key scheme is $(i+1)$-hop homomorphic w.r.t to a set of circuit families, then the constructed public-key scheme is i-hop homomorphic w.r.t to the same set.*

The discussion on the homomorphic properties of the scheme (i.e. the furthermore part) is presented in Section 2.5. To prove Theorem 2.3, we assume the existence of a homomorphic private-key scheme and use it to construct a public-key scheme (Construction 2.4). The main part of the proof is showing that this public-key scheme is indeed semantically secure.

Construction 2.4. *Let (G, E, D, H) be a homomorphic private-key scheme with respect to addition modulo 2 and let $m(\cdot)$ be the polynomial as in Definition 2.1. We denote by $H_\oplus$ the algorithm H when applied to the circuit family that computes addition modulo 2. The encryption scheme (G', E', D', H') is defined as follows:*

Key Generation - $G'(1^n)$: *Set $\ell = 10m(n)$. Select $k \leftarrow G(1^n)$, $X = (X_1, \ldots, X_\ell)$ and $Y = (Y_1, \ldots, Y_\ell)$ such that $X_i \leftarrow E_k(0)$ and $Y_i \leftarrow E_k(1)$ (with fresh random coins for each i). Output X, Y as the public-key and k as the private-key.*

Encryption - $E'_{X,Y}(\sigma)$: *Select a random subset $S \subseteq_R [\ell]$ that has size of parity σ (i.e. $|S| \equiv \sigma \bmod 2$) and output $H_\oplus(X_{\overline{S}} Y_S)$ (recall that $X_{\overline{S}} Y_S$ is a list of ℓ ciphertexts that are encryptions of 1 for coordinates in S and encryptions of 0 elsewhere).*

Decryption - $D'_k(c)$: *Output $D_k(c)$.*

Homomorphic Evalutation - $H'(C, (X, Y), c_0, \ldots, c_\ell)$**:** *Output $H(C, c_0, \ldots, c_\ell)$.*

We start by showing that the decryption algorithm correctly decrypts proper ciphertexts. We then proceed to the main part of the proof, showing that Construction 2.4 is indeed semantically secure. In Section 2.5 we discuss the homomorphic properties of the scheme.

Proposition 2.5. *For every $n \in \mathbb{N}$, $\sigma \in \{0,1\}$ and $((X,Y),k) \leftarrow G'(1^n)$:*

$$D'_k\left(E'_{X,Y}(\sigma)\right) = \sigma.$$

Proof. Based on the first property of homomorphic encryption (Definition 2.1),

$$D'_k\left(E'_{X,Y}(\sigma)\right) = D_k\left(H_\oplus\left(X_{\overline{S}}Y_S\right)\right) = \bigoplus_{i=1}^{\ell} D_k(C_i)$$

where $\oplus$ denotes addition modulo 2, $C_i = Y_i$ for $i \in S$ and $C_i = X_i$ otherwise. Since D decrypts correctly, $D_k(X_i) = 0$ and $D_k(Y_i) = 1$. Therefore, $D'_k\left(E'_{X,Y}(\sigma)\right) = \oplus_{i\in S} 1 = |S| \bmod 2 = \sigma$. □

We proceed to the main part of the proof, showing that Construction 2.4 is semantically secure.

Proposition 2.6. *If (G,E,D) is a multiple-message semantically secure private-key scheme then (G',E',D') is a semantically secure public-key scheme.*

Proof. Assume toward a contradiction that (G',E',D') is not semantically secure. This means that there exists a probabilistic polynomial-time adversary A' and a polynomial $p(\cdot)$ such that for infinitely many $n \in \mathbb{N}$:

$$\Pr_{\substack{(X,Y),k\leftarrow G'(1^n)\\ \sigma\in_R\{0,1\}}}\left[A'\left(X,Y,E'_{X,Y}(\sigma)\right)=\sigma\right] > \frac{1}{2} + \frac{1}{p(n)}\ . \tag{2.3}$$

To derive a contradiction, we consider n from this infinite set and construct a probabilistic polynomial-time adversary A for the underlying private-key scheme. A receives 2ℓ ciphertexts $(\alpha_1,\ldots,\alpha_\ell,\beta_1,\ldots,\beta_\ell)$ and will be shown to distinguish between the following two cases:

- $\alpha_1,\ldots,\alpha_\ell$ are encryptions of 0 and $\beta_1,\ldots,\beta_\ell$ are encryptions of 1.
- $\alpha_1,\ldots,\alpha_\ell,\beta_1,\ldots,\beta_\ell$ are encryptions of 0.

A operates as follows:

1. Set $X = (\alpha_1,\ldots,\alpha_\ell)$ and $Y = (\beta_1,\ldots,\beta_\ell)$.
2. Select $S \subseteq_R [\ell]$.
3. Output 1 if $A'(X,Y,H_\oplus(X_{\overline{S}}Y_S)) = |S| \bmod 2$ and 0 otherwise.

Accordingly,

$$\Pr_{\substack{k\leftarrow G(1^n)\\ \alpha_j,\beta_j}}\left[A\left(\alpha_1,\ldots,\alpha_\ell,\beta_1,\ldots,\beta_\ell\right)=1\right] = \Pr_{\substack{k\leftarrow G(1^n)\\ X,Y,S}}\left[A'\left(X,Y,H_\oplus\left(X_{\overline{S}}Y_S\right)\right) = |S| \bmod 2\right].$$

We proceed by analyzing A's behavior in the two different cases. In the first case, $\alpha_i = E_k(0)$ and $\beta_i = E_k(1)$. Consequently, $H_\oplus(X_{\overline{S}}Y_S)$ is distributed identically to an encryption of a random bit under E' and so, by Eq. (2.3), it holds that

$$\Pr_{\substack{k\leftarrow G(1^n)\\X,Y,S}}\left[A'\left(X,Y,H_\oplus\left(X_{\overline{S}}Y_S\right)\right)=|S| \bmod 2\right]=\Pr_{\substack{(X,Y),k\leftarrow G'(1^n)\\\sigma\in_R\{0,1\}}}\left[A'\left(X,Y,E'_{X,Y}\left(\sigma\right)\right)=\sigma\right]>\frac{1}{2}+\frac{1}{p(n)}\ .$$

In the second case, $\alpha_i = \beta_i = E_k(0)$. We argue that in this case for every $n \in \mathbb{N}$ and even for an unbounded adversary A',

$$\Pr_{\substack{k\leftarrow G(1^n)\\X,Y,S}}\left[A'\left(X,Y,H_\oplus\left(X_{\overline{S}},Y_S\right)\right)=|S| \bmod 2\right]<\frac{1}{2}+2^{-0.2\ell+m(n)+1}. \tag{2.4}$$

Equation (2.4) follows from an information-theoretic theorem (Theorem 2.7) that will be stated next and proved in Section 2.4.

Using Theorem 2.7, we conclude that A distinguishes between the two cases with non-negligible probability, in contradiction to the multiple-message security of (G, E, D), □

Information-Theoretic Theorem. Let Ω be a finite non-empty set and $\ell \in \mathbb{N}$. Let $\mu_1, \ldots, \mu_\ell$ be distributions over Ω and $\mu = \mu_1 \times \cdots \times \mu_\ell$ be a product distribution over Ω^ℓ. Let X and Y be independent random variables identically distributed according to μ over Ω^ℓ.

Theorem 2.7. *For any $\ell, m \in \mathbb{N}$ and any functions $h\colon \Omega^\ell \to \{0,1\}^m$ and $g\colon \Omega^\ell \times \Omega^\ell \times \{0,1\}^m \to \{0,1\}$, it holds that*

$$\Pr_{X,Y,S\subseteq_R[\ell]}\left[g\left(X,Y,h(X_{\overline{S}}Y_S)\right)=|S| \bmod 2\right]<\frac{1}{2}+2^{-0.2\ell+m+1}.$$

Equation (2.4) seems to follow immediately from Theorem 2.7 by setting A' as g, $H_\oplus$ as h and having X and Y distributed as ℓ independent encryptions of 0 each. However, there is a small subtlety - Theorem 2.7 addresses g and h that are *deterministic* functions, in contrast to A' and H that are *probabilistic* algorithms. Additionally, since X and Y are distributed w.r.t to the same randomly chosen key, they are not product distributions as required by Theorem 2.7.

Both issues are resolved by an averaging argument. If Eq. (2.4) does not hold for some $n \in \mathbb{N}$, then there exist random coins for A', H and a fixed private key k for which it does not hold. Once we fix these coins, A' and H become deterministic functions. Additionally, we set X and Y to each be distributed as ℓ encryptions of 0 under the fixed key k, which is in particular a product distribution. Thus, the hypothesis that Eq. (2.4) does not hold contradicts Theorem 2.7.

2.4 Proof of Theorem 2.7

Theorem 2.7 considers a game in which a computationally unbounded adversary sees X, Y and m bits of information on $X_{\overline{S}}Y_S$ and needs to decide whether S is of even or odd cardinality. In other words, the adversary specifies a function $h : \Omega^\ell \to \{0,1\}^m$ and based on $X, Y, h(X_{\overline{S}}Y_S)$ needs to find $|S| \bmod 2$. Theorem 2.7 states that winning this game with probability noticeably better than $\frac{1}{2}$ is impossible as long as m is sufficiently smaller than ℓ. Note that winning the game becomes easy if m is very large w.r.t ℓ[4] (as long as the probability of a collision in each coordinate, i.e. $\Pr[X_i = Y_i]$, is sufficiently small). Thus, we are interested in the case $m \ll \ell$.

Organization. The proof of Theorem 2.7 uses the Efron-Stein decomposition, an extension of Fourier analysis for general product distributions. We begin by presenting this decomposition, together with the relevant facts. We then turn to the actual proof of Theorem 2.7.

2.4.1 Efron-Stein Decomposition

Recall that X and Y are independent random variables identically distributed by μ, a product distribution over Ω^ℓ. We consider the inner-product space of functions from Ω^ℓ to $\mathbb{R}$, where the inner product of f and g is $\langle f, g\rangle \stackrel{\text{def}}{=} \mathbf{E}_X[f(X)g(X)]$. We stress that the expectation is over X (which is distributed according to μ). We use the convention that lowercase x and y refer to elements in Ω^ℓ (in contrast to uppercase X and Y which are random variables over Ω^ℓ).

Theorem 2.8 (Efron-Stein Decomposition [ES81])**.** *Any function $f : \Omega^\ell \to \mathbb{R}$ can be decomposed to $f = \sum_{S\subseteq[\ell]} f^S$, where $f^S : \Omega^\ell \to \mathbb{R}$ satisfy:*

1. *f^S only depends on the coordinates of x that reside in S (i.e. x_S).*
2. *For any $x \in \Omega^\ell$ and $S \not\supseteq U$ it holds that $E_Y[f^U(x_S Y_{\overline{S}})] = 0$.*

Note that if $\Omega = \{\pm 1\}$ it is easy to verify that the Fourier representation of the function is also its Efron-Stein decomposition (taking $f^S = \hat{f}(S)\chi_S$ where $\chi_S(x) = \prod_{i\in S} x_i$). In our general setting we denote $\hat{f}(S)^2 \stackrel{\text{def}}{=} \langle f^S, f^S\rangle$ (indeed, when $\Omega = \{\pm 1\}$ this notation agrees with the standard interpretation of $\hat{f}(S)$ in Fourier analysis of Boolean functions).

One of the important properties of this decomposition is that it is orthogonal and therefore Parseval's Equality holds.

Fact 2.9 (Orthogonality)**.** *For any $S \neq U$, f^S and f^U are orthogonal.*

[4] If $m \geq \ell \log(|\Omega|)$ just take h to be the identity function.

Proof. Assume without loss of generality that $S \not\supseteq U$. Since $X_S Y_{\overline{S}}$ is identically distributed to X,

$$\langle f^S, f^U \rangle = \mathop{\mathbf{E}}_{X}[f^S(X) f^U(X)] = \mathop{\mathbf{E}}_{X,Y}\left[f^S(X_S Y_{\overline{S}}) f^U(X_S Y_{\overline{S}})\right].$$

Based on the fact that f^S only depends on coordinates in S, we can replace $f^S(X_S Y_{\overline{S}})$ with $f^S(X_S X_{\overline{S}}) = f^S(X)$. Thus,

$$\langle f^S, f^U \rangle = \mathop{\mathbf{E}}_{X,Y}\left[f^S(X) f^U(X_S Y_{\overline{S}})\right] = \mathop{\mathbf{E}}_{X}\left[f^S(X) \mathop{\mathbf{E}}_{Y}\left[f^U(X_S Y_{\overline{S}})\right]\right].$$

But by the second property of the decomposition (Theorem 2.8), for every $x \in \Omega^\ell$, $\mathbf{E}_Y[f^U(x_S Y_{\overline{S}})] = 0$ and so we have $\langle f^S, f^U \rangle = 0$. □

Theorem 2.10 (Parseval's Equality)**.**

$$\sum_{S \subseteq [\ell]} \hat{f}(S)^2 = \mathop{\mathbf{E}}_{X}[f(X)^2].$$

Proof.

$$\sum_{S \subseteq [\ell]} \hat{f}(S)^2 = \sum_{S \subseteq [\ell]} \langle f^S, f^S \rangle = \sum_{S,T \subseteq [\ell]} \langle f^S, f^T \rangle = \langle \sum_{S \subseteq [\ell]} f^S, \sum_{T \subseteq [\ell]} f^T \rangle = \langle f, f \rangle,$$

where the second equality follows from orthogonality. □

The Efron-Stein decomposition has proved to be extremely useful in giving explicit expressions for the noise sensitivity of a function or the influence of a subset of its coordinates. We will use it to express the "stability" of a subset of coordinates, which is in a sense the complement of the influence for this set. The fact that we use is summarized in Proposition 2.11 (a similar analysis has been applied previously to give an explicit expression for influence, e.g., in [Bla09]).

Proposition 2.11. *If f is Boolean valued* (i.e. $f \colon \Omega^\ell \to \{0,1\}$)*, then for every $S \subseteq [\ell]$ it holds that:*

$$\Pr_{X,Y}[f(X) = f(X_{\overline{S}} Y_S) = 1] = \sum_{T \subseteq \overline{S}} \hat{f}(T)^2.$$

Proof. Using the fact that f is Boolean, the Efron-Stein decomposition, and linearity of expectation we have:

$$\begin{aligned}
\Pr_{X,Y}\left[f(X) = f(X_{\overline{S}} Y_S) = 1\right] &= \mathop{\mathbf{E}}_{X,Y}\left[f(X) f(X_{\overline{S}} Y_S)\right] \\
&= \mathop{\mathbf{E}}_{X,Y}\left[\sum_{T \subseteq [\ell]} f^T(X) \sum_{U \subseteq [\ell]} f^U(X_{\overline{S}} Y_S)\right] \\
&= \sum_{U,T \subseteq [\ell]} \mathop{\mathbf{E}}_{X}\left[f^T(X) \mathop{\mathbf{E}}_{Y}\left[f^U(X_{\overline{S}} Y_S)\right]\right]. \qquad (2.5)
\end{aligned}$$

From the Efron-Stein decomposition we have that if $U \not\subseteq \overline{S}$ then $E_Y[f^U(X_{\overline{S}}Y_S)] = 0$, whereas if $U \subseteq \overline{S}$ then $E_Y[f^U(X_{\overline{S}}Y_S)] = f^U(X)$. Thus, Eq. (2.5) yields that:

$$\Pr_{X,Y}[f(X) = f(X_{\overline{S}}Y_S) = 1] = \sum_{T\subseteq[\ell]}\sum_{U\subseteq\overline{S}} \mathop{\mathbf{E}}_{X}[f^T(X)f^U(X)] = \sum_{T\subseteq[\ell]}\sum_{U\subseteq\overline{S}}\langle f^T, f^U\rangle = \sum_{T\subseteq\overline{S}}\langle f^T, f^T\rangle$$

where the last equality follows from orthogonality. □

2.4.2 Proof of Theorem 2.7

We would like to show that for a *typical* $\gamma \in \{0,1\}^m$, the number of odd S that map to γ (that is $h(X_{\overline{S}}Y_S) = \gamma$) and the number of even such S are roughly the same. This would imply that any adversary, which sees only X, Y and γ, cannot guess whether γ was produced from an odd or even S, which is exactly what we are looking to prove. To formalize this, we introduce the following notation; for $\gamma \in \{0,1\}^m$, we define:

$$I_{\text{odd}}(X,Y,\gamma) \stackrel{\text{def}}{=} |\{\, T \subseteq [\ell] : h(X_{\overline{T}}Y_T) = \gamma \text{ and } |T| \text{ is odd}\,\}| \tag{2.6}$$

$$I_{\text{even}}(X,Y,\gamma) \stackrel{\text{def}}{=} |\{\, T \subseteq [\ell] : h(X_{\overline{T}}Y_T) = \gamma \text{ and } |T| \text{ is even}\,\}| \tag{2.7}$$

Organization. We begin by presenting some basic facts. The proof will be composed of two lemmas, Lemma 2.14 (which is the main lemma) states that for *every* $\gamma \in \{0,1\}^m$, w.h.p, the number of odd T that map to γ is fairly close to the number of even T (in absolute terms). Lemma 2.18 states that for a *typical* γ the total number of T that map to it is very large. Combining these two lemmas we prove Theorem 2.7.

2.4.2.1 Basic Facts

We first present two basic facts that follow immediately from the structure of $X_{\overline{S}}Y_S$.

Fact 2.12. *For every $\gamma \in \{0,1\}^m$, there exists a constant $\mu_\gamma \in [0,1]$ such that for every $S \subseteq [\ell]$:*

$$\Pr_{X,Y}[h(X_{\overline{S}}Y_S) = \gamma] = \mu_\gamma.$$

Proof. Define $\mu_\gamma \stackrel{\text{def}}{=} \Pr[h(X) = \gamma]$ and note that $\Pr[h(X_{\overline{S}}Y_S) = \gamma] = \Pr[h(X) = \gamma]$ (because $X_{\overline{S}}Y_S$ and X are identically distributed). □

Fact 2.13. *For every $S, T \subseteq [\ell]$ and $\gamma \in \{0,1\}^m$,*

$$\Pr_{X,Y}[h(X_{\overline{S}}Y_S) = h(X_{\overline{T}}Y_T) = \gamma] = \Pr_{X,Y}\left[h(X) = h(X_{\overline{S\oplus T}}Y_{S\oplus T}) = \gamma\right]$$

where $S \oplus T$ denotes the symmetric difference of two sets, i.e., $S \oplus T \stackrel{def}{=} (S\backslash T) \cup (T\backslash S)$.

Proof. Using the fact that $X_{\overline{S}}Y_S$ is identically distributed to X, we can swap Y_S and X_S in the expression $\Pr[h(X_{\overline{S}}Y_S) = h(X_{\overline{T}}Y_T)]$. Hence, $X_{\overline{S}}Y_S$ becomes X. For $X_{\overline{T}}Y_T$ we use X for coordinates that are in $\overline{T}\backslash S$ or in $T \cap S$ and use Y for coordinates that are in $\overline{T} \cap S$ or in $T\backslash S$. Therefore, $X_{\overline{T}}Y_T$ becomes $X_{\overline{S\oplus T}}Y_{S\oplus T}$. □

2.4.2.2 The Main Lemma

Lemma 2.14. *For every $\gamma \in \{0,1\}^m$, it holds that:*

$$\Pr_{X,Y}\left[|I_{odd}(X,Y,\gamma) - I_{even}(X,Y,\gamma)| \geq 2^{0.6\ell}\right] \leq 2^{-0.2\ell}.$$

Throughout the proof of this lemma, in all probabilistic statements, the probability is always over X and Y. Additionally, since X and Y are clear from the context, we use the shorthand $I_{\text{odd}}(\gamma)$ (resp. $I_{\text{even}}(\gamma)$) for $I_{\text{odd}}(X,Y,\gamma)$ (resp. $I_{\text{even}}(X,Y,\gamma)$).

Foreseeing that we will prove Lemma 2.14 by an application of Chebyshev's inequality, we proceed by bounding the expectation and variance of $I_{\text{odd}}(\gamma) - I_{\text{even}}(\gamma)$.

Proposition 2.15. *For every $\gamma \in \{0,1\}^m$, it holds that:*

$$\mathbf{E}[I_{odd}(\gamma) - I_{even}(\gamma)] = 0.$$

Proof. $I_{\text{odd}}(\gamma)$ can be expressed as a sum of indicator variables: $I_{\text{odd}}(\gamma) = \sum_{\text{odd } T} I_T(\gamma)$, where $I_T(\gamma)$ is an indicator for the event $h(X_{\overline{T}}Y_T) = \gamma$. Thus,

$$\mathbf{E}\left[I_{\text{odd}}(\gamma)\right] = \mathbf{E}\left[\sum_{\text{odd } T} I_T(\gamma)\right] = \sum_{\text{odd } T} \mathbf{E}\left[I_T(\gamma)\right] = \sum_{\text{odd } T} \Pr\left[h(X_{\overline{T}}Y_T) = \gamma\right] = 2^{\ell-1}\mu_\gamma$$

where the last equality follows from Fact 2.12. Similarly, it is easy to see that $\mathbf{E}\left[I_{\text{even}}(\gamma)\right] = 2^{\ell-1}\mu_\gamma$ and thus $\mathbf{E}\left[I_{\text{odd}}(\gamma) - I_{\text{even}}(\gamma)\right] = 0$. □

Proposition 2.16. *For every $\gamma \in \{0,1\}^m$, it holds that*

$$\mathbf{Var}[I_{odd}(\gamma) - I_{even}(\gamma)] \leq 2^\ell.$$

Proof. Recall that I_{odd} and I_{even} can be expressed as the sum of the indicator variables I_T (as defined in the proof of Proposition 2.15). Thus, using Proposition 2.15 and some manipulations we have:

$$\begin{aligned}
\mathbf{Var}\left[I_{\text{odd}}(\gamma) - I_{\text{even}}(\gamma)\right] &= \mathbf{E}\left[\left(I_{\text{odd}}(\gamma) - I_{\text{even}}(\gamma)\right)^2\right] \\
&= \mathbf{E}\left[I_{\text{odd}}(\gamma)^2\right] + \mathbf{E}\left[I_{\text{even}}(\gamma)^2\right] - 2\,\mathbf{E}\left[I_{\text{odd}}(\gamma)I_{\text{even}}(\gamma)\right] \\
&= \mathbf{E}\left[\left(\sum_{\text{odd } T} I_T(\gamma)\right)^2\right] + \mathbf{E}\left[\left(\sum_{\text{even } T} I_T(\gamma)\right)^2\right] \\
&\quad - 2\,\mathbf{E}\left[\left(\sum_{\text{odd } T} I_T(\gamma)\right)\left(\sum_{\text{even } T} I_T(\gamma)\right)\right] \\
&= \sum_{\substack{T,U\subseteq[\ell] \text{ s.t.} \\ |T|=|U| \bmod 2}} \mathbf{E}\left[I_T(\gamma)I_U(\gamma)\right] - \sum_{\substack{T,U\subseteq[\ell] \text{ s.t.} \\ |T|\neq|U| \bmod 2}} \mathbf{E}\left[I_T(\gamma)I_U(\gamma)\right] \\
&= \sum_{T,U} (-1)^{|T \oplus U|}\,\mathbf{E}\left[I_T(\gamma)I_U(\gamma)\right] \\
&= \sum_{T,U} (-1)^{|T \oplus U|}\Pr\left[h(X_{\overline{T}}Y_T) = h(X_{\overline{U}}Y_U) = \gamma\right].
\end{aligned}$$

Now using Fact 2.13 we have:

$$\begin{aligned}\mathbf{Var}[I_{\text{odd}}(\gamma) - I_{\text{even}}(\gamma)] &= \sum_{T,U}(-1)^{|T \oplus U|} \Pr\left[h(X) = h(X_{\overline{T \oplus U}} Y_{T \oplus U}) = \gamma\right] \\ &= \sum_{T,U}(-1)^{|T|} \Pr\left[h(X) = h(X_{\overline{T}} Y_T) = \gamma\right] \\ &= 2^\ell \sum_{i=0}^{\ell}(-1)^i \sum_{T\colon |T|=i} \Pr\left[h(X) = h(X_{\overline{T}} Y_T) = \gamma\right].\end{aligned}$$

Let $f\colon \Omega^\ell \to \{0,1\}$ be the indicator function for $h(X) = \gamma$. Clearly, for every T, it holds that $\Pr\left[h(X) = h(X_{\overline{T}} Y_T) = \gamma\right] = \Pr\left[f(X) = f(X_{\overline{T}} Y_T) = 1\right]$ and so by using Proposition 2.11 we derive:

$$\begin{aligned}\mathbf{Var}[I_{\text{odd}}(\gamma) - I_{\text{even}}(\gamma)] &= 2^\ell \sum_{i=0}^{\ell}(-1)^i \sum_{T\colon |T|=i} \Pr\left[f(X) = f(X_{\overline{T}} Y_T) = 1\right] \\ &= 2^\ell \sum_{i=0}^{\ell}(-1)^i \sum_{T\colon |T|=i} \left(\sum_{U \subseteq \overline{T}} \hat{f}(U)^2\right) \\ &= 2^\ell \sum_{i=0}^{\ell}(-1)^i \sum_{R\colon |R|=\ell-i} \left(\sum_{U \subseteq R} \hat{f}(U)^2\right).\end{aligned}$$

Note that each $\hat{f}(U)^2$ in the sum appears $\binom{\ell-|U|}{i}$ times with respect to each i (and this holds even when $i > \ell - |U|$). Thus:

$$\begin{aligned}\mathbf{Var}[I_{\text{odd}}(\gamma) - I_{\text{even}}(\gamma)] &= 2^\ell \sum_{i=0}^{\ell}(-1)^i \sum_{U \subseteq [\ell]} \binom{\ell - |U|}{i} \hat{f}(U)^2 \\ &= 2^\ell \sum_{U \subseteq [\ell]} \hat{f}(U)^2 \sum_{i=0}^{\ell}(-1)^i \binom{\ell - |U|}{i} \\ &= 2^\ell \sum_{U \subseteq [\ell]} \hat{f}(U)^2 (1-1)^{\ell - |U|} \\ &= 2^\ell \hat{f}([\ell])^2.\end{aligned}$$

Finally, using Parseval's Equality (Theorem 2.10) and the fact that range of f is $\{0,1\}$:

$$\mathbf{Var}[I_{\text{odd}}(\gamma) - I_{\text{even}}(\gamma)] = 2^\ell \hat{f}([\ell])^2 \leq 2^\ell \sum_{S \subseteq [\ell]} \hat{f}(S)^2 = 2^\ell \mathop{\mathbf{E}}_X[f(X)^2] \leq 2^\ell.$$

□

Deriving Lemma 2.14. Applying Chebyshev's inequality, while using Propositions 2.15 and 2.16, we get that

$$\Pr\left[|I_{\text{odd}}(\gamma) - I_{\text{even}}(\gamma)| \geq 2^{0.6\ell}\right] \leq \frac{\mathbf{Var}[I_{\text{odd}}(\gamma) - I_{\text{even}}(\gamma)]}{2^{1.2\ell}} \leq \frac{2^{\ell}}{2^{1.2\ell}} = 2^{-0.2\ell}.$$

□

2.4.2.3 Completing the Proof

Lemma 2.14 addresses the case where γ is fixed. However, we need to handle γ that are chosen according to a specific distribution ($\gamma \sim h(X_{\overline{S}}Y_S)$). Since we consider such γ, it is convenient to define:

$$\tilde{I}_{\text{even}}(X,Y,S) = I_{\text{even}}\left(X,Y,h(X_{\overline{S}}Y_S)\right) \tag{2.8}$$

$$\tilde{I}_{\text{odd}}(X,Y,S) = I_{\text{odd}}\left(X,Y,h(X_{\overline{S}}Y_S)\right) \tag{2.9}$$

$$\Delta_{X,Y}(S) = \left|\tilde{I}_{\text{even}}(X,Y,S) - \tilde{I}_{\text{odd}}(X,Y,S)\right| \tag{2.10}$$

Corollary 2.17.

$$\Pr_{X,Y,S\subseteq_R[\ell]}\left[\Delta_{X,Y}(S) \geq 2^{0.6\ell}\right] \leq 2^{-0.2\ell+m}.$$

Proof. If $\Delta_{X,Y}(S) \geq 2^{0.6\ell}$ then for $\gamma = h(X_{\overline{S}}Y_S)$ it holds that $|I_{\text{odd}}(X,Y,\gamma) - I_{\text{even}}(X,Y,\gamma)| \geq 2^{0.6\ell}$. Thus:

$$\Pr_{X,Y,S}\left[\Delta_{X,Y}(S) \geq 2^{0.6\ell}\right] \leq \Pr_{X,Y}\left[\exists\, \gamma \in \{0,1\}^m \text{ s.t. } |I_{\text{odd}}(X,Y,\gamma) - I_{\text{even}}(X,Y,\gamma)| \geq 2^{0.6\ell}\right].$$

The corollary follows by applying a union bound and Lemma 2.14. □

Consider all $T \subseteq [\ell]$ that map (via h) to the same value as S. Corollary 2.17 bounds the difference between the number of even and odd such T. However, since it does so only in absolute terms, it is meaningless if the number of such T is small. Lemma 2.18 shows that for a typical γ, w.h.p, this is not the case.

Notation. Recall our convention that lowercase x and y refer to elements in Ω^{ℓ}. For fixed x and y, we define $I_{x,y}(\gamma)$ to be the total number of $T \subseteq [\ell]$ that h maps to γ, i.e.,

$$I_{x,y}(\gamma) \stackrel{\text{def}}{=} I_{\text{odd}}(x,y,\gamma) + I_{\text{even}}(x,y,\gamma) = \left|\{\, T \subseteq [\ell] :\ h(x_{\overline{T}}y_T) = \gamma \,\}\right|. \tag{2.11}$$

Since we are sometimes interested in typical γ's, we also define

$$\tilde{I}_{x,y}(S) \stackrel{\text{def}}{=} I_{x,y}\left(h(x_{\overline{S}}, y_S)\right). \tag{2.12}$$

Lemma 2.18. *For every $x, y \in \Omega^{\ell}$,*

$$\Pr_{S}\left[\tilde{I}_{x,y}(S) \leq 2^{0.8\ell}\right] \leq 2^{-0.2\ell+m}.$$

Proof.

$$\begin{aligned}
\Pr_S\left[\tilde{I}_{x,y}(S) \le 2^{0.8\ell}\right] &= \sum_{\gamma\in\{0,1\}^m} \Pr_S\left[\tilde{I}_{x,y}(S) \le 2^{0.8\ell} \bigwedge h(x_{\overline{S}}y_S) = \gamma\right] \\
&= \sum_{\gamma\in\{0,1\}^m} \Pr_S\left[I_{x,y}(\gamma) \le 2^{0.8\ell} \bigwedge h(x_{\overline{S}}y_S) = \gamma\right] \\
&= \sum_{\gamma:\ I_{x,y}(\gamma)\le 2^{0.8\ell}} \Pr_S\left[h(x_{\overline{S}}y_S) = \gamma\right] \\
&\le 2^m \cdot \frac{2^{0.8\ell}}{2^\ell}\ .
\end{aligned}$$

□

Lemma 2.18 together with Corollary 2.17 imply, that w.h.p, $\tilde{I}_{\text{odd}}(X,Y,S)$ and $\tilde{I}_{\text{even}}(X,Y,S)$ are very close (since their sum is big and their difference is small). Intuitively, this implies that an adversary that tries to find $|S| \bmod 2$ from X, Y and $h(X_{\overline{S}}Y_S)$ can not do much better than a fair coin toss. Proposition 2.19 formalizes this intuitive connection.

Proposition 2.19. *For every $x, y \in \Omega^\ell$:*

$$\Pr_S[g\left(x,y,h(x_{\overline{S}}y_S)\right) = |S| \bmod 2] \le \frac{1}{2} + \frac{1}{2}\cdot \mathbf{E}\left[\frac{\Delta_{x,y}(S)}{\tilde{I}_{x,y}(S)}\right]$$

where $\Delta_{x,y}(S)$ and $\tilde{I}_{x,y}(S)$ are as defined in Eq. (2.10) *and Eq.* (2.12) *respectively.*

Proof. Since x and y are fixed, and we quantify over *all* g and h, we can just consider functions that depend on x and y. Thus, we denote $g_{x,y}(\gamma) \stackrel{\text{def}}{=} g(x,y,\gamma)$ and $h_{x,y}(S) \stackrel{\text{def}}{=} h(x_{\overline{S}}y_S)$.

Choosing a random subset $S \subseteq [\ell]$ is equivalent to first choosing $\gamma = h_{x,y}(S)$ and then choosing uniformly over all $T \subseteq [\ell]$ that h maps to γ. Formally, let S be a uniformly distributed subset of $[\ell]$ and let T_S be distributed uniformly over $\{\, T \subseteq [\ell] :\ h_{x,y}(T) = h_{x,y}(S)\,\}$. Since S and T_S are identically distributed (by the uniform distribution) it holds that

$$\begin{aligned}
\Pr_S\left[g_{x,y}\left(h_{x,y}(S)\right) = |S| \bmod 2\right] &= \Pr_{S,T_S}\left[g_{x,y}\left(h_{x,y}(S)\right) = |T_S| \bmod 2\right] \\
&= \mathbf{E}_S\left[\Pr_{T_S}\left[g_{x,y}\left(h_{x,y}(S)\right) = |T_S| \bmod 2\right]\right].
\end{aligned}$$

For fixed S, by definition, $\Pr_{T_S}\left[g_{x,y}(h_{x,y}(S)) = |T_S| \bmod 2\right]$ is just

$$\frac{|\{\, T :\ (|T| \bmod 2) = g_{x,y}(h_{x,y}(S)) \text{ and } h_{x,y}(T) = h_{x,y}(S)\,\}|}{|\{\, T :\ h_{x,y}(T) = h_{x,y}(S)\,\}|}.$$

The numerator of this expression equals the number of T's that map to the same value as S whose size is of some fixed parity (note that $g_{x,y}(h_{x,y}(S))$ is fixed) and thus is at

most $\max\left(\tilde{I}_{\text{odd}}(x,y,S), \tilde{I}_{\text{even}}(x,y,S)\right)$. Likewise, the denominator is exactly $\tilde{I}_{x,y}(S)$ and so we have:

$$\begin{aligned}\Pr_S\left[g_{x,y}\left(h_{x,y}(S)\right) = |S| \bmod 2\right] &\leq \mathop{\mathbf{E}}_S\left[\frac{\max\left(\tilde{I}_{\text{odd}}(x,y,S), \tilde{I}_{\text{even}}(x,y,S)\right)}{\tilde{I}_{x,y}(S)}\right] \\ &= \frac{1}{2} + \frac{1}{2}\cdot\mathop{\mathbf{E}}_S\left[\frac{\Delta_{x,y}(S)}{\tilde{I}_{x,y}(S)}\right].\end{aligned}$$

□

Deriving Theorem 2.7. Corollary 2.17 and Lemma 2.18 imply that:

$$\Pr_{X,Y,S\subseteq_R[\ell]}\left[\frac{\Delta_{X,Y}(S)}{\tilde{I}_{X,Y}(S)} < 2^{-0.2\ell}\right] > 1 - 2\cdot 2^{-0.2\ell+m}.$$

Therefore,

$$\mathop{\mathbf{E}}_{X,Y,S\subseteq_R[\ell]}\left[\frac{\Delta_{X,Y}(S)}{\tilde{I}_{X,Y}(S)}\right] < \left(1 - 2^{-0.2\ell+m+1}\right)\cdot 2^{-0.2\ell} + 2^{0.2\ell+m+1}\cdot 1 < 2^{-0.2\ell+m+2}.$$

And so, by Proposition 2.19,

$$\Pr_{X,Y,S\subseteq_R[\ell]}\left[g\left(X,Y,h(X_{\overline{S}}Y_S)\right) = |S| \bmod 2\right] < \frac{1}{2} + 2^{-0.2\ell+m+1}.$$

□

2.5 Homomorphic Properties of the Public-Key Scheme

In this section, we discuss the homomorphic properties of the public-key scheme presented in Construction 2.4. Specifically, we shall show that if the private-key scheme supports $i+1$ repeated homomorphic operations then the public-key scheme supports i such operations. Intuitively, this follows by the fact that the public-key encryption algorithm applies a single homomorphic operation (see Fact 2.21).

Proposition 2.20. *Suppose G, E, D, H are an $(i+1)$-hop homomorphic private-key scheme w.r.t to a set of circuit families $\mathcal{C}$ that includes addition modulo 2. Then G', E', D', H' as defined in Construction 2.4 are an i-hop homomorphic public-key scheme w.r.t the set $\mathcal{C}$.*

Theorem 2.3 shows that (G', E', D', H') is indeed a public-key encryption scheme and so, we only need to show that the scheme supports i repeated evaluations of circuits from $\mathcal{C}$.

Let $(X,Y), k$ be a pair of encryption/decryption keys of the public scheme (w.r.t to the security parameter n). We denote the j-th level ciphertexts of the private-key scheme by $W_j(k)$ and the j-th level ciphertexts of the public-key scheme by $W'_j(X,Y)$.

Fact 2.21. *For every $j \in \mathbb{N}$, $W'_j(X,Y) \subseteq W_{j+1}(k)$.*

Proof. By induction on j. □

Let $\{C_k\}_k \in \mathcal{C}$, $0 \leq j \leq i$, $\ell = \ell(n)$ and $w_1, \ldots, w_\ell$ be j-th level ciphertexts of the public-key scheme (i.e., in $W'_j(X,Y)$). We proceed by showing that the first property of Definition 2.2 (Eq. 2.1) holds. By Fact 2.21, it holds that $w_1, \ldots, w_\ell \in W_{j+1}(k)$ and thus,

$$\begin{aligned} H'(C_\ell, (X,Y), w_1, \ldots, w_\ell) &= H(C_\ell, w_1, \ldots, w_\ell) \\ &= C_\ell(D_d(w_1), \ldots, D_d(w_\ell)) \quad = C_\ell(D'_d(w_1), \ldots, D'_d(w_\ell)). \end{aligned}$$

where the first and third equalities follow from the definition of H' and D' respectively and the second equality follows from the first requirement of Definition 2.2, noting that $w_1, \ldots, w_\ell$ are ciphertexts of level $j+1 \leq i+1$ of the private-key scheme.

A similar argument shows that the second property of Definition 2.2 (Eq. 2.2) holds. Indeed, since $w_1, \ldots, w_\ell \in W'_j(X,Y) \subseteq W_{j+1}(k)$ it holds that,

$$|H'(C_\ell, (X,Y), w_1, \ldots, w_\ell)| = |H(C_\ell, w_1, \ldots, w_\ell)| \leq m(n)$$

for every $0 \leq j \leq i$.

Chapter 3

A Taxonomy of Enhanced Trapdoor Permutations

3.1 Introduction

A collection of trapdoor permutations is a collection of efficiently computable permutations that are hard to invert on the average, with the additional property, that each permutation has a trapdoor that makes the permutation easy to invert. Trapdoor permutations are among the most fundamental primitives of cryptography and have been used to construct a variety of schemes and protocols, most notably, public-key encryption [Yao82] and signature schemes [BM92].

Trapdoor permutations were also believed to imply oblivious transfer and (efficient prover) non-interactive zero-knowledge proofs for $\mathcal{NP}$ but it seems that some additional structure is required for these applications (see, e.g. [Gol04, Gol09]). The point is that in these applications, the adversary may get auxiliary information such as the randomness used to sample an image of the permutation, and this auxiliary information may allow inverting the permutation or approximating its hardcore predicate. The phenomenon was first observed in the context of constructing oblivious transfer (see [Gol04]) and later in the context of non-interactive zero-knowledge proofs (see [Gol09]). This led to the introduction of enhanced and doubly-enhanced trapdoor permutations, which suffice for the construction of oblivious transfer and non-interactive zero-knowledge proofs for $\mathcal{NP}$, respectively. These phenomena motivate further study of the hardness requirements from enhanced trapdoor permutations, and on closer examination still more issues arise. For example, while enhanced trapdoor permutations do suffice for one-out-of-two oblivious transfer, we show that the standard construction (as in [Gol04]) needs to be adapted in order to obtain one-out-of-k oblivious transfer, for $k \geq 3$. The motivating question behind this work is asking what added features are necessary and sufficient for applications such as oblivious transfer and non-interactive zero-knowledge proofs for $\mathcal{NP}$.

Our approach is to define a number of abstract scenarios, where each scenario captures the type of information available to the adversary and the information that we wish to keep secret. Each scenario leads to a corresponding notion of an enhanced trapdoor per-

mutations which is hard to invert in that scenario. We study the relations between these variants of enhanced trapdoor permutations as well as the relation to the aforementioned applications, while noting that all these variants of enhanced trapdoor permutations are implied by the doubly-enhanced property.

3.1.1 Trapdoor Permutations

Loosely speaking, a collection of one-way permutations is a collection of efficiently computable permutations that are hard to invert on the average, i.e., given a random permutation f from the collection and a random element x from its domain, it is infeasible to find $f^{-1}(x)$. Each permutation is represented by an index α and has an associated domain D_α over which it is defined. A natural domain to consider is $\{0,1\}^{|\alpha|}$; however, this is not necessarily the case in general, and our only requirement is that it is possible to efficiently sample elements (uniformly) from this domain. We denote the domain sampler by S, and use the convention that $S(\alpha; r)$ refers to the (deterministic) output of S on input α and the random string r.

A collection of trapdoor permutations (TDP) is a collection of one-way permutations $\{f_\alpha \colon D_\alpha \to D_\alpha\}_\alpha$ for which, each permutation has an associated trapdoor that makes the permutation easy to invert. We require an efficient way to generate an index together with the corresponding trapdoor.

Enhanced TDP. Consider an adversary for inverting a TDP that is given not only an element x but also the random coins that were used to sample x. Goldreich [Gol04] noted that there are TDPs that can be inverted in such a setting (assuming that TDPs exist)[1] and showed that this issue captures a real security concern in the standard protocol for oblivious transfer (which was believed to work based on any TDP). He therefore defined the notion of an *enhanced* TDP which is a TDP that is infeasible to invert even given the random coins that sampled the element to be inverted. That is, given an index of a permutation α and a random string r it is infeasible to compute $x \stackrel{\text{def}}{=} f_\alpha^{-1}(S(\alpha; r))$.

Hardcore Predicates. A hardcore predicate of a TDP is an efficiently computable predicate, defined over the domain of the permutation that is infeasible to approximate, given only the image of the element. In other words, given α and x it is easy to compute $h(x)$ but given only α and $f_\alpha(x)$ it is infeasible to approximate $h(x)$. Note that since f_α is a permutation, $h(x)$ is information theoretically determined by α, $f_\alpha(x)$ and therefore, $h(x)$ is only hard to approximate in a computational sense. An *enhanced* hardcore predicate of an *enhanced* TDP is naturally defined w.r.t the enhanced security property. That is, based on α and r, it is infeasible to approximate $h(x)$ where $x = f_\alpha^{-1}(S(\alpha; r))$. Goldreich and Levin [GL89] showed a hardcore predicate that holds for (a minor modification of)

[1] Given any TDP, consider changing its sampling algorithm S to $S'(\alpha; r) \stackrel{\text{def}}{=} f_\alpha(S(\alpha; r))$. The random coins of S' always give away the preimage under f_α of sampled elements.

any TDP. If the TDP is enhanced then the Goldreich-Levin predicate is an *enhanced* hardcore predicate.

3.1.2 Are Enhanced Hardcore Bits Pseudorandom?

Let $\{f_\alpha \colon D_\alpha \to D_\alpha\}_\alpha$ be a standard TDP with a hardcore predicate h. Suppose that we are given a randomly selected index of a permutation α and two random elements from its domain $y_1, y_2 \in D_\alpha$. By definition, it is infeasible to compute the hardcore bit of the inverse of any single element (i.e. $h(f_\alpha^{-1}(y_j))$. Moreover, intuitively it seems as though these two bits are pseudorandom, and in particular it is infeasible to compute any *relationship* between these two bits.

A simple argument shows that this is indeed the case. To prove this, assume toward a contradiction that there exists an algorithm A that given α, y_1 and y_2 computes $h(f_\alpha^{-1}(y_1)) \oplus h(f_\alpha^{-1}(y_2))$, where $\oplus$ denotes exclusive-or. We use A to construct an adversary A' for the hardcore predicate h. Recall that A' is given α and y_1 and needs to compute $h(f_\alpha^{-1}(y_1))$. The key point is that A' can generate $h(f_\alpha^{-1}(y_2)), y_2$ by itself[2] and then invoke A on α, y_1, y_2 to obtain $b = h(f_\alpha^{-1}(y_1)) \oplus h(f_\alpha^{-1}(y_2))$. Finally, using $h(f_\alpha^{-1}(y_2))$, the adversary A' outputs $b \oplus h(f_\alpha^{-1}(y_2))$ which indeed equals $h(f_\alpha^{-1}(y_1))$.

Surprisingly perhaps, this argument does not extend to the enhanced setting. Suppose that $\{f_\alpha\}$ is an *enhanced* TDP with a domain sampler S and an *enhanced* hardcore predicate h. Given α, r_1 and r_2, is it feasible to compute $h(x_1) \oplus h(x_2)$ where $x_j = f_\alpha^{-1}(S(\alpha; r_j))$? In fact, this may indeed be feasible. The key point, which causes the extension of the proof from the standard setting to the enhanced setting to fail, is that it may not be feasible to generate a sample of the form $(h(x_2), r_2)$ without using the trapdoor. In Appendix A we present an enhanced trapdoor permutation based on quadratic residuosity for which this is the case. In Section 3.3 we use this property to show that the standard one-out-of-k oblivious transfer protocol is insecure for $k \geq 3$.

Using the equivalence of pseudorandomness and unpredictability, enhanced hardcore bits may also be predictable in the following sense. Given α, $(r_1, h(x_1))$ and r_2 it may be feasible to predict $h(x_2)$. The types of scenarios that we consider in this work are generalizations of this attack. That is, scenarios in which the adversary is given samples (e.g. r_1 together with $h(x_1)$) that it may not be able to generate by itself and is required to execute a task that is infeasible without the samples (e.g. compute $h(x_2)$ based on r_2). For each scenario we consider a corresponding variant of an enhanced trapdoor permutation that is hard in that scenario. We consider connections between these variants while distinguishing between hardness that holds w.r.t a fixed number of samples and hardness that holds w.r.t any (polynomial) number of samples.

3.1.3 Organization

We start in Section 3.2 by presenting the formal definitions of trapdoor permutations, hardcore predicates and oblivious transfer. In Section 3.3, we demonstrate the type

[2] By selecting $x \leftarrow S(\alpha)$ and outputting $h(x), f_\alpha(x)$.

of problem encountered when using enhanced trapdoor permutations by analyzing the standard OT protocol. In Section 3.4, we discuss the aforementioned scenarios in which enhanced trapdoor permutations are not hard to invert or have hardcore predicates that are not hard to predict.

3.2 Definitions

A function $\epsilon\colon \mathbb{N} \to [0,1]$ is *negligible* if for every polynomial $p(\cdot)$ and all sufficiently large $n \in \mathbb{N}$ it holds that, $\epsilon(n) < \frac{1}{p(n)}$. We use $\text{neg}(n)$ and $\text{poly}(n)$ to respectively denote some unspecified negligible function and polynomial. Throughout this manuscript all polynomials are assumed to be positive.

3.2.1 Collections of Trapdoor Permutations

Formally, we define a collection of trapdoor permutations (TDP) as follows:

Definition 3.1. *A* TDP *is a collection of permutations* $\{f_\alpha : D_\alpha \to D_\alpha\}_\alpha$ *together with the following associated probabilistic polynomial-time algorithms:*

1. *An* index sampler I *that given the security parameter* 1^n*, outputs an index of a permutation, denoted* α *and a corresponding trapdoor, denoted* τ*.*
2. *A* domain sampler S *that on input* α *(the index of a permutation), outputs a uniformly distributed element* $x \in D_\alpha$*.*
3. *An* evaluation algorithm F *(for Forward) that, given* α *and* x*, computes the value of the permutation* f_α *on* x*, i.e., outputs* $f_\alpha(x)$*.*
4. *An* inverting algorithm B *(for Backward) that, given the trapdoor of the permutation* τ *and an element* x*, inverts the permutation on* x*, i.e., outputs* $f_\alpha^{-1}(x)$*.*

The security requirement is that for every probabilistic polynomial-time algorithm A*,*

$$\Pr_{\substack{(\alpha,\tau)\leftarrow I(1^n)\\ x\leftarrow S(\alpha)}}\left[A(\alpha,x) = f_\alpha^{-1}(x)\right] = \text{neg}(n) \tag{3.1}$$

where the probability is also over the coin tosses of A*.*

An enhanced TDP is one for which it is infeasible to invert elements even given the random coins that sampled them:

Definition 3.2. *A TDP* $\{f_\alpha\colon D_\alpha \to D_\alpha\}$ *with domain sampler* S*, is* enhanced *if it for every probabilistic polynomial-time algorithm* A*,*

$$\Pr_{\substack{(\alpha,\tau)\leftarrow I(1^n)\\ r\leftarrow\{0,1\}^{poly(n)}}}\left[A(\alpha,r) = f_\alpha^{-1}(S(\alpha;r))\right] = \text{neg}(n) \tag{3.2}$$

where once again, the probability is also over the coin tosses of A*.*

A hardcore predicate is an efficiently computable predicate defined over the domain of a TDP, that is infeasible to compute based only on an image of an element:

Definition 3.3. *Let $\{f_\alpha \colon D_\alpha \to D_\alpha\}_\alpha$ be a TDP. The predicate h, defined over the domain of the permutations, is a* hardcore predicate *if it can be computed efficiently and for every probabilistic polynomial-time algorithm A,*

$$\Pr_{\substack{(\alpha,\tau)\leftarrow I(1^n)\\ x\leftarrow S(\alpha)}}\left[A(x) = h(f_\alpha^{-1}(x))\right] = \frac{1}{2} + neg(n) \tag{3.3}$$

An enhanced hardcore predicate is defined analogously, allowing the adversary access to the random string that sampled the element:

Definition 3.4. *Let $\{f_\alpha \colon D_\alpha \to D_\alpha\}_\alpha$ be a TDP with domain sampler S. The hardcore predicate h is* enhanced *if for every probabilistic polynomial-time algorithm A,*

$$\Pr_{\substack{(\alpha,\tau)\leftarrow I(1^n)\\ r\leftarrow\{0,1\}^{poly(n)}}}\left[A(r) = h(f_\alpha^{-1}(S(\alpha;r)))\right] = \frac{1}{2} + neg(n) \tag{3.4}$$

Goldreich and Levin [GL89] showed that if $\{f_\alpha\}_\alpha$ is a trapdoor permutation then the trapdoor permutation $g_\alpha(x,s) = (f_\alpha(x), s)$ where $|x| = |s|$, has a hardcore predicate $h(x,s) = \langle x,s\rangle = \sum x_i s_i \bmod 2$. If $\{f_\alpha\}_\alpha$ is an enhanced TDP then h is an *enhanced* hardcore predicate of $\{g_\alpha\}_\alpha$.

3.2.2 Oblivious Transfer

One-out-of-k oblivious transfer (OT) is an interactive protocol consisting of two parties, a sender S and a receiver R. The input of S is composed of k-bits $\sigma_1, \ldots, \sigma_k$ and the input of R is an index $i \in [k]$. At the end of the protocol, the receiver, R, should output σ_i but learn nothing about the sender's input other than σ_i and the sender, S, should learn nothing about the receiver's input (i.e., i). These privacy requirements should hold in a computational sense, with respect to a security parameter n, (which is given to both parties in unary). We restrict our attention to the "semi-honest" model. In this model, each party acts according to the protocol but may write down anything it sees. We mention that a protocol in the "semi-honest" model can be compiled to a protocol that is secure against malicious adversaries by using zero-knowledge proofs (see [Gol04]).

This formulation of OT was introduced by Even et-al [EGL85]. A three-message protocol for OT based on enhanced trapdoor permutations was given by [EGL85, GMW87]. We refer to this protocol (or actually to its description in [Gol04]) as the standard OT protocol. The standard OT protocol uses an enhanced TDP $\{f_\alpha \colon D_\alpha \to D_\alpha\}_\alpha$ with corresponding algorithms I, S, F, B (recall that I is the index/trapdoor sampler, S is the domain sampler, F computes the permutation and B inverts it using the trapdoor) and an enhanced hardcore predicate h (e.g. the Goldreich-Levin hardcore predicate [GL89]). The protocol is depicted in Figure 3.1.

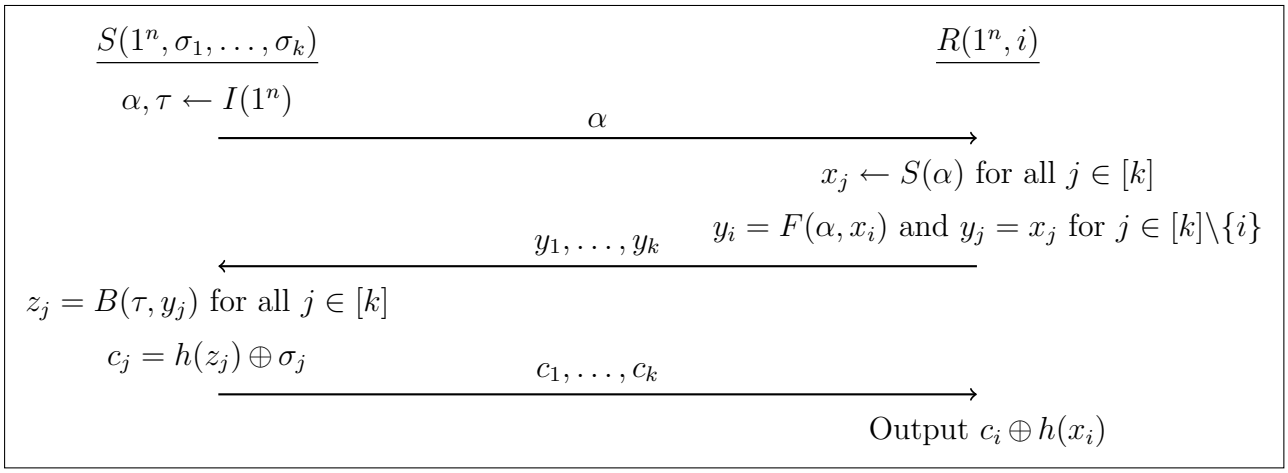

Figure 3.1: One-out-of-k Oblivious Transfer.

To formalize the semi-honest model, we use the notion of the view of each player. The view of player P with respect to security parameter n is a random variable $View_P\left(i, (\sigma_1, \ldots, \sigma_k)\right)$ which consists of everything player P sees in the interaction between R on input i and S on input $\sigma_1, \ldots, \sigma_k$, including its own random coin tosses and the received messages. Using this notion we define an OT protocol as follows:

Definition 3.5. *Let $k \geq 2$ be a natural number. (S, R) are a one-out-of-k oblivious transfer (OT) protocol if S and R are interactive probabilistic polynomial-time algorithms and it holds that:*

1. *(Correctness) For every $\sigma_1, \ldots, \sigma_k \in \{0,1\}$ and $i \in [k]$, when $R(1^n, i)$ interacts with $S(1^n, \sigma_1, \ldots, \sigma_k)$, it holds that R outputs σ_i and S outputs nothing.*

2. *(Sender Privacy) There exists a probabilistic polynomial-time simulator Sim_R such that for every $\sigma_1, \ldots, \sigma_k \in \{0,1\}$ and $i \in [k]$, the ensembles $\{Sim_R(i, \sigma_i)\}_{n \in \mathbb{N}}$ and $\{View_R(i, (\sigma_1, \ldots, \sigma_k))\}_{n \in \mathbb{N}}$ are computationally indistinguishable.*

3. *(Receiver Privacy) There exists a probabilistic polynomial-time simulator Sim_S such that for every $\sigma_1, \ldots, \sigma_k \in \{0,1\}$ and $i \in [k]$, the ensembles $\{Sim_S(\sigma_1, \ldots, \sigma_k)\}_{n \in \mathbb{N}}$ and $\{View_S(i, (\sigma_1, \ldots, \sigma_k))\}_{n \in \mathbb{N}}$ are computationally indistinguishable.*

If the output of Sim_S (resp. Sim_R) is identically distributed to the actual view of the sender (resp. receiver) then we say that the receiver (resp. sender) has perfect privacy.

3.3 Failure of the one-out-of-k OT Protocol for $k \geq 3$

In this section we show that the standard OT protocol fails for $k \geq 3$. We start of by proving that it is indeed correct for $k = 2$. We then proceed to show the problem that arises when trying to extend this proof to $k = 3$, or larger k.

3.3.1 The Case $k = 2$

Recall that the standard protocol (Figure 3.1) is based on an enhanced TDP $\{f_\alpha\}_\alpha$ with corresponding algorithms I, S, F, B and an enhanced hardcore predicate h.

When both parties follow the standard protocol the receiver outputs σ_i, thus the protocol is indeed correct. The (prefect) privacy of the receiver (in the semi-honest model) is also immediate and follows from the fact that f_α is a permutation. We note that correctness and the privacy of the receiver hold for any $k \geq 2$.

The privacy of the sender is less trivial. For sake of simplicity we assume that $i = 1$ and consider an interaction between the receiver, R, and the sender, S given the input σ_1, σ_2. The view of the receiver is:

$$(i = 1, \sigma_1), (r_1, r_2), \left(\alpha, h(S(\alpha; r_1)) \oplus \sigma_1, h(f_\alpha^{-1}(S(\alpha; r_2))) \oplus \sigma_2\right).$$

To prove that privacy of the sender holds, we need to present a simulator for this view. The simulator $Sim_R(i = 1, \sigma_1)$ chooses $(\alpha, \tau) \leftarrow I(1^n)$, samples $r_1, r_2 \leftarrow \{0,1\}^{\mathrm{poly}(n)}$ and outputs:

$$(i = 1, \sigma_1), (r_1, r_2), \left(\alpha, h(S(\alpha; r_1)) \oplus \sigma_1, h(f_\alpha^{-1}(S(\alpha; r_2)))\right).$$

Note that the only difference between the actual view and the output of the simulator is in the last element. However, using the fact that h is an *enhanced* hardcore predicate, $r_2, h(f_\alpha^{-1}(S(\alpha; r_2)))$ and $r_2, h(f_\alpha^{-1}(S(\alpha; r_2))) \oplus \sigma_2$ are computationally indistinguishable, which in turn implies that the actual view is computationally indistinguishable from the output of the simulator.

3.3.2 The Case $k = 3$

Consider an attempt to extend the proof for the case $k = 2$ to the case $k = 3$. Once again, for simplicity, we assume $i = 1$. The natural extension of the proof is to have the simulator output $h(f_\alpha^{-1}(S(\alpha; r_2)))$ and $h(f_\alpha^{-1}(S(\alpha; r_3)))$ instead of the actual received message $h(f_\alpha^{-1}(S(\alpha; r_2))) \oplus \sigma_2$ and $h(f_\alpha^{-1}(S(\alpha; r_3))) \oplus \sigma_3$. The problem is that it may be easy to distinguish the output of the simulator from the actual received message, using the property described in Section 3.1.2.

We stress that it is not only the natural extension of the proof to the case $k = 3$ that fails, and the protocol is indeed insecure. To see this (again assuming $i = 1$) recall that R should learn σ_1 but nothing about σ_2 and σ_3. However, based on the protocol R learns $(r_2, b_2 \oplus \sigma_2)$ and $(r_3, b_3 \oplus \sigma_3)$ (where $b_i = h(f_\alpha^{-1}(S(\alpha; r_i)))$). Given that R can also compute $b_2 \oplus b_3$ from r_1 and r_2, it can easily compute $\sigma_1 \oplus \sigma_2$, contradicting the supposed privacy of the sender.

3.3.3 Fixing the Protocol

One way to fix the standard OT protocol is by using the well known (simple) reduction from general k to $k = 2$. As shown in Section 3.3.1, one-out-of-two OT can be based on *any* enhanced TDP, hence, the following holds:

Theorem 3.6. *If there exists an enhanced TDP then for any $k \geq 2$, there exists a protocol for one-out-of-k OT.*

An alternate approach, that considers the original protocol, is shown in Section 3.4.2.2.

3.4 Problematic Scenarios for Enhanced Trapdoor Permutations

In this section, we discuss different scenarios in which it may be insecure to use enhanced TDPs. We start by presenting these scenarios and the corresponding TDPs and proceed to show connections between these variants of enhanced TDPs.

Throughout this section, we consider an enhanced TDP $\{f_\alpha \colon D_\alpha \to D_\alpha\}_\alpha$ with corresponding algorithms I, S, F, B and a hardcore predicate h. By α we denote a random index from this collection (with respect to a security parameter n). For $j \in \mathbb{N}$, we use r_j to denote uniformly distributed random coins for the sampling algorithm S and x_j to denote the inverse of the corresponding sampled element, i.e., $x_j \stackrel{\text{def}}{=} f_\alpha^{-1}(S(\alpha, r_j))$. b_j denotes the corresponding hardcore bit, i.e., $b_j \stackrel{\text{def}}{=} h(x_j)$.

3.4.1 The Scenarios

The attack we presented on the OT protocol in Section 3.3 was based on the existence of an enhanced TDP with the property that given α, r_1 and r_2, it is feasible to compute $b_1 \oplus b_2$. This means that an adversary that is given a sample of the form (r_1, b_1), can compute the hardcore predicate, i.e., given r_2 (in addition to α and (r_1, b_1)) the adversary can compute $b_2 = h(f_\alpha^{-1}(S(\alpha; r_2)))$. We generalize this type of attack and consider the following scenarios:

1. **Scenario BX:** Based on α and i samples of the form $(r_1, b_1), \ldots, (r_i, b_i)$ it is feasible to invert the permutation, i.e., from r_{i+1} compute $x_{i+1} = f_\alpha^{-1}(S(\alpha; r_{i+1})$.

2. **Scenario BB:** Based on α and i samples of the form $(r_1, b_1), \ldots, (r_i, b_i)$ it is feasible to break the hardcore predicate, i.e., from r_{i+1}, compute $b_{i+1} = h(x_{i+1})$.

3. **Scenario XX:** Based on α and i samples of the form $(r_1, x_1), \ldots, (r_i, x_i)$ it is feasible to invert the permutation, i.e., compute x_{i+1} as in Scenario 1.

4. **Scenario XB:** Based on α and i samples of the form $(r_1, x_1), \ldots, (r_i, x_i)$ it is feasible to break the hardcore predicate, i.e., compute b_{i+1} as in Scenario 2.

Scenarios 1-4 are respectively referred to as Scenarios BX, BB, XX, XB, where the convention is that the first letter represents what the adversary is given (B for hardcore bits and X for preimages) and the second letter represents the goal of the adversary (B to approximate the hardcore bit and X to invert the permutation).

A few immediate relations between the scenarios are depicted in Figure 3.2, where an arrow from Scenario x to Scenario y means that an adversary in the setting of Scenario x implies an adversary in Scenario y. These relations follow from the fact that b_j can be efficiently computed from x_j. Hardness holds in the opposite direction, that is, an arrow from scenario x to scenario y means that a TDP that is hard in the setting of scenario y is also hard in the setting of scenario x.

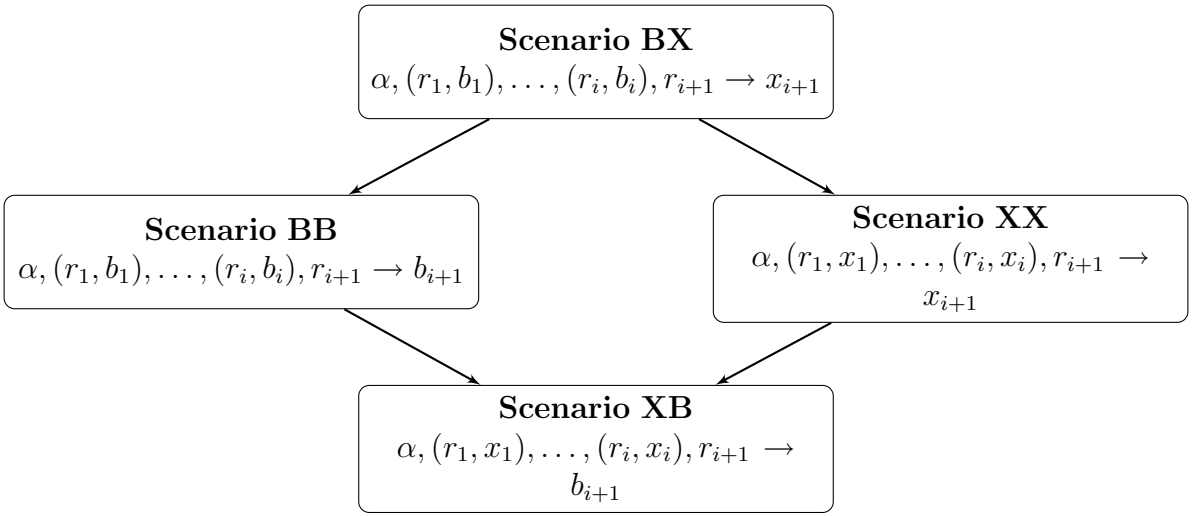

Figure 3.2: Attacks on Enhanced Trapdoor Permutations.

Connection to Doubly-Enhanced TDP Scenario XB is actually the setting of the protocol for non-interactive zero-knowledge proofs for $\mathcal{NP}$. In this protocol, the verifier is presented with samples of the form (r_j, x_j). To prove that the protocol is zero-knowledge, we need to argue that the verifier cannot compute the hardcore predicate, however, if the TDP is vulnerable to an attack in the setting of Scenario XB then the hardcore predicate becomes easy to compute. Goldreich [Gol09], addressed the problem raised there by defining *doubly-enhanced* TDPs and showing that they are sufficient. Recall that a doubly-enhanced TDP is defined as follows:

Definition 3.7. *Let $\{f_\alpha \colon D_\alpha \to D_\alpha\}_\alpha$ be an enhanced TDP with domain sampling algorithm S. This collection is* doubly-enhanced *if there exists a probabilistic polynomial-time algorithm that on input α outputs a pair (r, x) such that r is uniformly distributed as random coins of S and $f_\alpha(x) = S(\alpha; r)$.*

Thus, the "doubly-enhanced" property guarantees the ability to generate samples of the form (r_j, x_j) (from which can be derived also samples of the form (r_j, b_j)). This implies, in particular, that a doubly-enhanced TDP is hard in all the above scenarios w.r.t to polynomially many samples. This follows from the fact that any adversary that requires samples (r_j, x_j) or (r_j, b_j) can be converted to an adversary that does not, by simply generating the necessary samples itself (using the algorithm guaranteed by Definition 3.7).

3.4.2 Hardness of Enhanced TDP w.r.t a Fixed Number of Samples

In this section, and the following one, we show connections between the variants of enhanced TDP that correspond to the scenarios discussed above. We distinguish between hardness that holds for a *fixed* number of samples (discussed in this section) and hardness that holds for polynomially may samples, for *any* polynomial (discussed in Section 3.4.3).

The results presented in this section are depicted in Figure 3.3. Each box represents a TDP that is hard in one of the scenarios. Arrows represent connections between these primitives. A solid arrow between from primitive X to Y means that "X is Y"[3] whereas a dotted arrow means that a transformation is required, that is, if there exists X then there exists Y. Some of the arrows are labeled with further restrictions, e.g., "GL", which means that the result holds when using the GL hardcore predicate. Note that all downward pointing arrows follow from the fact that the hardcore bit of an element is efficiently computable.

3.4.2.1 Scenario BX

Recall that in Scenario BX, the adversary needs to invert the permutation based on samples of the form (r_j, b_j). We first show that *any* enhanced TDP with *any* enhanced hardcore predicate is hard to invert in Scenario BX if at most logarithmically many samples (r_j, b_j) are revealed to the adversary. We proceed by showing that by modifying an enhanced TDP, we can actually obtain an enhanced TDP that is hard to invert even given *polynomially* many samples.

Theorem 3.8. *Let $\{f_\alpha\}_\alpha$ be an enhanced TDP with an enhanced hardcore predicate h. Then, $\{f_\alpha\}$ is hard to invert in the setting of Scenario BX for $i = O(\log n)$ samples.*

Proof. We use an adversary A for Scenario BX to construct an adversary A' that inverts the permutation in the enhanced setting (i.e. an adversary that inverts based on the random string of the sampling algorithm). The first observation is that A' can enumerate all possible values for logarithmically many hardcore bits. For each sequence of values of hardcore bits, A' runs A to produce a candidate preimage. The second observation is that it is possible to verify the result; that is, A' can check whether a candidate element is indeed the preimage. Details follow.

Assume toward a contradiction that there exists a probabilistic polynomial-time algorithm A that given $\alpha, (r_1, b_1), \ldots, (r_{i(n)}, b_{i(n)}), r_{i(n)+1}$ outputs $x_{i(n)+1}$ with non-negligible probability. We use A to invert the permutation in the enhanced setting, contradicting the assumption that it is an *enhanced* trapdoor permutation.

Given α and r we want to find $x = f_\alpha^{-1}(S(\alpha; r))$ where S is the domain sampling algorithm of $\{f_\alpha\}$. To do this, we select $r_1, \ldots, r_{i(n)}$ as $i(n)$ uniformly distributed random strings of the sampling algorithm S. We then enumerate over all possible values of

[3] We assume that all TDP are by default in the form required for GL. Thus, we do not view the (minor) modification required for the GL hardcore predicate as a transformation.

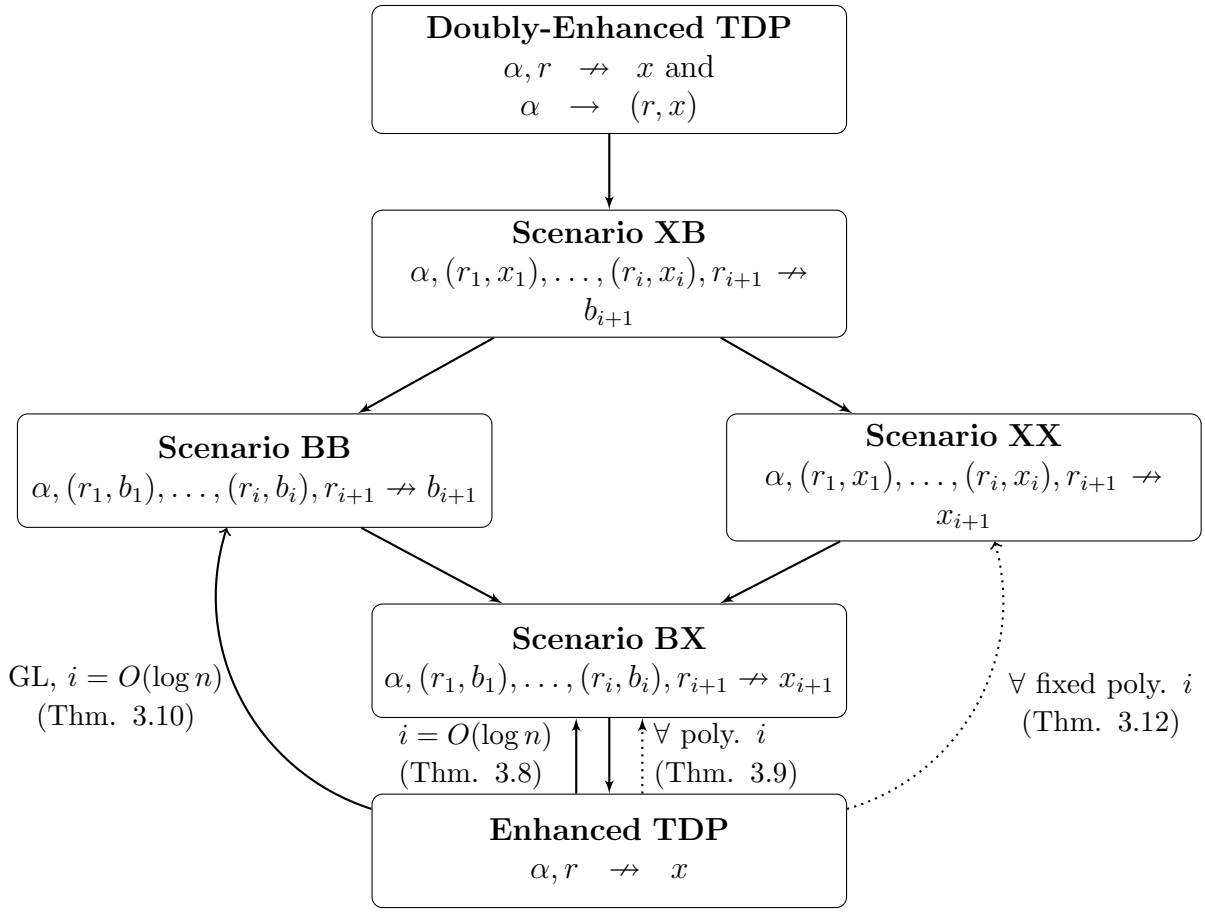

Figure 3.3: Hardness of Enhanced TDP w.r.t a *fixed* number of samples.

hardcore bits of $r_1, \ldots, r_{i(n)}$ by going over all $c_1, \ldots, c_{i(n)} \in \{0,1\}$. For each choice of $c_1, \ldots, c_{i(n)}$ we run $A(\alpha, (r_1, c_1), \ldots, (r_{i(n)}, c_{i(n)}), r)$ and obtain a candidate x'. For each candidate x', we check whether $f_\alpha(x') = S(\alpha; r)$ and output x' if this is the case. Note that after a polynomial number of iterations we will reach the correct choice of hardcore bits and then A inverts $S(\alpha; r)$ with non-negligible probability. □

Theorem 3.8 states that any enhanced TDP is hard to invert based on logarithmically many samples in Scenario BX. Indeed, this holds for any enhanced TDP and no modification is required. If we allow modifications, then as shown by the following theorem, we can actually obtain an enhanced TDP that is hard to invert even given polynomially many samples.

Theorem 3.9. *Let $\{f_\alpha \colon D_\alpha \to D_\alpha\}_\alpha$ be an enhanced TDP with a hardcore predicate h_f. The direct product of $\{f_\alpha\}$ with itself, denoted $\{g_{\alpha,\beta} : D_\alpha \times D_\beta \to D_\alpha \times D_\beta\}$ and defined as $g_{\alpha,\beta}(x, y) = (f_\alpha(x), f_\beta(y))$, is an enhanced TDP with an enhanced hardcore predicate $h_g(x, y) = h_f(y)$ that is hard to invert in Scenario BX w.r.t any polynomial $i(\cdot)$.*

Proof. We denote the domain sampling algorithm of $\{f_\alpha\}$ (resp. $\{g_{\alpha,\beta}\}$) by S_f (resp. S_g). Note that $S_g((\alpha,\beta);(r,r')) \stackrel{\text{def}}{=} (S_f(\alpha;r), S_f(\beta,r'))$. We denote the random string of S_g by $r'' = (r,r')$ where r is used to sample the first element (from D_α) and r' is used for the second one (from D_β).

Suppose there exists an adversary A that inverts $\{g_{\alpha,\beta}\}$ based on a polynomial number of samples (r''_j, b''_j), where $r''_j = (r_j, r'_j)$ and $b''_j = h_g\left(g^{-1}_{\alpha,\beta}\left(S_g\left((\alpha,\beta);r''_j\right)\right)\right) = h_f\left(f^{-1}_\beta\left(S_f\left(\beta;r'\right)\right)\right) = b'_j$. We use A to construct an adversary A' that inverts $\{f_\alpha\}$. The key point is that it is possible to generate the necessary samples for A using only the trapdoor of β. Thus, to invert f_α, we generate β together with the corresponding trapdoor. We use this trapdoor to generate samples (r''_j, b''_j) where $b''_j = b'_j = h_f\left(f^{-1}_\beta\left(S_f\left(\beta;r'\right)\right)\right)$ and invoke A to invert $g_{\alpha,\beta}$ and in particular f_α. Details follow.

Given α and r, the adversary A' needs to find $x = f^{-1}_\alpha\left(S_f\left(\alpha;r\right)\right)$. This is done by first sampling an index β together with the corresponding trapdoor. Consider the permutation $g_{\alpha,\beta}$. For this permutation, it is easy to generate samples (r''_j, b''_j), where $r''_j = (r_j, r'_j)$ and $b''_j = h_f\left(f^{-1}_\beta\left(S_f\left(\beta;r'_j\right)\right)\right)$ since the hardcore predicate depends only β, which we can invert. Thus, to invert f_α, we invoke A on the index (α,β), $i(n)$ samples (r''_j, b_j) and the random coins $r'' = (r,r')$ (where r is the random string given as input and r' is an independent random string). By our assumption, with non-negligible probability, the adversary outputs a preimage $x'' = (x,x')$ of $S_g\left((\alpha,\beta);(r,r')\right) = S_f\left(\alpha;r\right), S_f\left(\beta;r'\right)$. In particular we have $x = f^{-1}_\alpha\left(S_f\left(\alpha;r\right)\right)$ and so A' outputs x. □

3.4.2.2 Scenario BB

Recall that Scenario BB is the setting that causes the standard OT protocol to fail for $k \geq 3$ (see Section 3.3). We show that the Goldreich-Levin (GL) hardcore predicate [GL89], is unpredictable in this setting as long as at most $i = O(\log n)$ samples (r_j, b_j) are revealed to the adversary. Thus, when implemented using the GL hardcore predicate, the standard OT protocol is secure for $k = O(\log n)$.

Theorem 3.10. *Let $\{f_\alpha \colon D_\alpha \to D_\alpha\}_\alpha$ be an enhanced TDP and assume for simplicity that all elements in D_α are of length n. Let $\{g_\alpha \colon D_\alpha \times \{0,1\}^n \to D_\alpha \times \{0,1\}^n\}_\alpha$ be the enhanced TDP defined as $g_\alpha(x,s) = f_\alpha(x), s$ where $|x| = |s| = n$ and let $h(x,s) \stackrel{def}{=} \langle x,s\rangle = \sum_{i=1}^n x_i s_i \bmod 2$ be the GL hardcore predicate of g. Then h is unpredictable in the setting of Scenario BB for $i = O(\log n)$ samples.*

We denote the domain sampling algorithm of $\{f_\alpha\}$ (resp. $\{g_\alpha\}$) by S_f (resp. S_g). Note that $S_g(\alpha);(r,s)) \stackrel{\text{def}}{=} (S_f(\alpha;r), s)$.

To proof Theorem 3.10, we show that given α and i random strings $(r_1,s_1),\ldots,(r_i,s_i)$ of S_g, the sampling algorithm of $\{g_\alpha\}_\alpha$, it is infeasible to approximate $\bigoplus_{j\in U} b_j$, for any non-empty set $U \subseteq [i]$ (where $b_j = h(x_j,s_j)$ and $x_j = f^{-1}_\alpha(S(\alpha;r_j))$). The theorem follows by applying the computational XOR lemma[4] for hardcore functions [Gol01, Lemma

[4]This lemma shows that if it is infeasible to compute the parity of a random subset of logarithmically many hardcore bits, then they are pseudorandom.

2.5.8], which holds for $i = O(\log n)$, and the equivalence of pseudorandomness and unpredictability. Thus it suffices to prove the following:

Proposition 3.11. *Let $i \stackrel{def}{=} i(n)$ be a polynomial. For any adversary A, any polynomial $p(\cdot)$, all sufficiently large n and any non-empty set $U \subseteq [i(n)]$:*

$$\Pr_{\substack{\alpha,\tau \leftarrow I(1^n) \\ (r_1,s_1),\ldots,(r_i,s_i) \leftarrow \{0,1\}^{poly(n)} \times \{0,1\}^{poly(n)}}} \left[A\left(\alpha, (r_1,s_1), \ldots, (r_i,s_i)\right) = \bigoplus_{j \in U} b_j \right] = \frac{1}{2} + \frac{1}{p(n)} \quad (3.5)$$

where $b_j \stackrel{def}{=} h(x_j, s_j) = \langle x_j, s_j \rangle$ and $x_j \stackrel{def}{=} f_\alpha^{-1}(S(\alpha; r_j))$.

Proof. Assume toward a contradiction that this is not the case. That is, there exists an infinite set of n, a set $U = \{j_1, \ldots, j_{\ell(n)}\}$, and an adversary A that computes $\bigoplus_{j \in U} b_j$ based on α and $(r_1, s_1), \ldots, (r_i, s_i)$. The main observation is that $\bigoplus_{j \in U} b_j = \bigoplus_{j \in U} \langle x_j, s_j \rangle = \langle x_{j_1} \circ \cdots \circ x_{j_{\ell(n)}}, s_{j_1} \circ \cdots \circ s_{j_{\ell(n)}} \rangle$ where $\circ$ denotes concatenation.

As a mental experiment, consider the trapdoor permutation $\{f'_\alpha \colon D^\ell_\alpha \to D^\ell_\alpha\}$ defined as $f'_\alpha(x_1, \ldots, x_{\ell(n)}) = (f_\alpha(x_1), \ldots, f_\alpha(x_{\ell(n)}))$. Using the sampling algorithm $S_{f'}(\alpha; r_1, \ldots, r_{\ell(n)}) = S(\alpha; r_1), \ldots, S(\alpha, r_{\ell(n)})$, the collection $\{f'_\alpha\}$ is in fact an *enhanced* trapdoor permutation[5]. If we apply the GL modification to $\{f'_\alpha\}$ we obtain the enhanced trapdoor permutation $g'_\alpha(x_1, \ldots, x_{\ell(n)}, s_1, \ldots, s_{\ell(n)}) = f_\alpha(x_1), \ldots, f_\alpha(x_{\ell(n)}), s_1, \ldots, s_{\ell(n)}$ with the enhanced hardcore predicate $\langle x_1 \circ \cdots \circ x_{\ell(n)}, s_1 \circ \cdots \circ s_{\ell(n)} \rangle$. By definition of an enhanced hardcore predicate, this means that given $\alpha, r_1, \ldots, r_{\ell(n)}, s_1, \ldots, s_{\ell(n)}$ it is infeasible to approximate $\langle x_1 \circ \cdots \circ x_{\ell(n)}, s_1 \circ \cdots \circ s_{\ell(n)} \rangle$ in contradiction to our assumption. □

3.4.2.3 Scenario XX

In this scenario, the adversary is given an index α of a permutation, i samples of the form (r_j, x_j), and an additional random string r_{i+1} and needs to invert the permutation on $S(\alpha; r_{i+1})$, i.e., compute x_{i+1}. We show how to transform any enhanced TDP to one that is hard to invert in the setting of Scenario XX *as long as the number of revealed samples is known ahead of time*, that is, first the number of revealed samples is fixed and then we construct a TDP that is hard to invert w.r.t this number of samples. A disadvantage of our technique is that the length of the index increases linearly with the number of samples. In fact, we can only construct an enhanced TDP that is hard to invert given $m^{1-\epsilon}$ samples where m is the length of the new index (for any constant $\epsilon > 0$).

Theorem 3.12. *If there exists an enhanced TDP, then for every polynomial $q(\cdot)$, there exists an enhanced TDP that is hard to invert in the setting of Scenario XX with respect to $q(n)$ samples.*

Proof. Let $\{f_\alpha\}_\alpha$ be an enhanced TDP with corresponding algorithm I, S, F, B.

[5] If from $\alpha, r_1, \ldots, r_{\ell(n)}$ it is feasible to compute $x_1, \ldots, x_{\ell(n)}$ then it particular it is feasible to compute x_1 from α, r_1.

Construction 3.13. *We construct an enhanced TDP f' with algorithms I', S', F', B' that is hard to invert in Scenario XX with $q(n)$ samples:*

$I'(1^n)$**:** *Invoke $I(1^n)$, the original index sampler, $2q(n) \cdot n$ times to obtain a $(2q(n) \times n)$-sized matrix of indexes $\underline{\alpha} \stackrel{def}{=} \{\alpha_{i,j}\}_{i \in [2q(n)], j \in [n]}$ and a corresponding $(2q(n) \times n)$-sized matrix of trapdoors $\underline{\tau} = \{\tau_{i,j}\}_{i \in [2q(n)], j \in [n]}$. Output $\underline{\alpha}$ as the index and $\underline{\tau}$ as the trapdoor.*

$S'(\underline{\alpha})$**:** *From each column $j \in [n]$ of the matrix $\underline{\alpha}$, select at random an entry $s_j \in_R [2q(n)]$ and sample an element from the corresponding permutation's domain, $x_j \leftarrow S(\alpha_{s_j,j})$. Output $(s_1, \ldots, s_n, x_1, \ldots, x_n)$.*

$F'(\underline{\alpha}, (s_1, \ldots, s_n, x_1, \ldots, x_n))$**:** *For every $j \in [n]$, compute the permutation $\alpha_{s_j,j}$ on x_j by invoking $y_j = F(\alpha_{s_j,j}, x_j)$. Output $(s_1, \ldots, s_n, y_1, \ldots, y_n)$.*

$B'(\underline{\tau}, (s_1, \ldots, s_n, y_1, \ldots, y_n))$**:** *For every $j \in [n]$, invert the permutation $\alpha_{s_j,j}$ on y_j by invoking $x_j = B(\tau_{s_j,j}, y_j)$. Output $(s_1, \ldots, s_n, x_1, \ldots, x_n)$.*

Using the fact that $\{f_\alpha\}$ is a TDP, $\{f'_{\underline{\alpha}}\}$ forms a collection of permutations and S' samples elements uniformly from the domain, as required. Furthermore, using the trapdoor $\underline{\tau}$, it is easy to invert $f'_{\underline{\alpha}}$.

We show that an A' adversary that inverts $\{f'_{\underline{\alpha}}\}$ in the setting of Scenario XX can be used to construct an adversary A that inverts $\{f_\alpha\}$ in the enhanced setting. Recall that A is given an index α and a random string r and needs to find x s.t. $x = f_\alpha^{-1}(S(\alpha; r))$. We first sketch the high-level idea of the proof and then go into details.

First, A generates an index matrix $\underline{\alpha}$ together with the corresponding trapdoor matrix $\underline{\tau}$. Then, A selects $q(n) + 1$ random strings for the sampling algorithm S'. Note that each random string specifies a single permutation from each column of $\underline{\alpha}$. The first $q(n)$ random strings will be used to construct samples for A', and the last random string will be used (after a modification) as the challenge for A'.

The key point is that for each column of $\underline{\alpha}$, with probability $\frac{q(n)}{2q(n)} = \frac{1}{2}$, there exists an entry that is not used by any of the first $q(n)$ random strings. Thus, with probability $1 - 2^{-n}$, one of the n indexes specified by the last random string was not specified by any of the first $q(n)$ samples. After finding the coordinate (i, j) of such an index in the matrix $\underline{\alpha}$ (or halting if it does not exist), A replaces the j-th block of the last random string by r and the (i, j)-th entry of $\underline{\alpha}$ by α. Since none of the first $q(n)$ random strings use α, the adversary A can invert them to obtain the required $q(n)$ samples for A'. If A' is successful then in particular it inverts $S(\alpha; r)$, hence obtaining the required preimage.

We proceed to describe the proof in detail. Assume toward a contradiction that there exists a probabilistic polynomial-time adversary A' that inverts $\{f'_{\underline{\alpha}}\}$ with non-negligible probability based on $q(n)$ samples. Thus, A' inverts $f_{\underline{\alpha}}$ based on $\underline{\alpha}$ and $q(n)$ samples of the form $\left(s_1^{(k)}, \ldots, s_n^{(k)}, x_1^{(k)}, \ldots, x_n^{(k)}\right), \left(r_1^{(k)}, \ldots, r_n^{(k)}\right)$ (for all $k \in [q(n)]$) where $x_j^{(k)}$ is the inverse of the element sampled by $r_j^{(k)}$ w.r.t the permutation $\alpha_{s_j^{(k)},j}$. To simplify notation,

we denote $\alpha(k,j) \stackrel{\text{def}}{=} \alpha_{s_j^{(k)},j}$. We use A' to construct an adversary A that on input α and r computes $f_\alpha^{-1}(S(\alpha;r))$ and operates as follows:

1. For every $k \in [q(n)]$, select $s_1^{(k)},\ldots,s_n^{(k)} \in_R [2q(n)]$.
2. Select $s'_1,\ldots,s'_n \in_R [2q(n)]$.
3. Find $t \in [n]$ such that $s'_t \notin \{s_t^{(1)},\ldots,s_t^{(q(n))}\}$. If no such t exists, halt.
4. Sample an index matrix $\underline{\alpha} = \{\alpha_{i,j}\}_{i\in[2q(n)],j\in[n]}$ together with the corresponding trapdoor $\underline{\tau}$ by invoking $I'(1^n)$. Replace $\alpha_{s'_t,t}$ with α ($\tau_{s'_t,t}$ is irrelevant and can be erased).
5. For $k \in [q(n)]$:
   (a) Select $r_1^{(k)},\ldots,r_{q(n)}^{(k)}$ as uniformly distributed random coins for S.
   (b) For $j \in [n]$, set $x_j^{(k)} = f^{-1}_{\alpha(k,j)}(S(\alpha(k,j);r_j^k))$.
6. Select n uniformly distributed random strings $r'_1,\ldots,r'_n$ for S. Replace r'_t with r.
7. Invoke A' on index $\underline{\alpha}$, the samples $\left(s_1^{(k)},\ldots,s_n^{(k)},x_1^{(k)},\ldots,x_n^{(k)}\right), \left(r_1^k,\ldots,r_n^{(k)}\right)$ (for all $k \in [q(n)]$) and the challenge $(s'_1,\ldots,s'_n,r'_1,\ldots,r'_n)$. The output of A should be $(s'_1,\ldots,s'_n,x'_1,\ldots,x'_n)$, halt if this is not the case.
8. Output x'_t.

We first argue that A is indeed efficient. The only step that appears problematic is inverting in step 5b however this can be done efficiently since we only invert permutations for which we have the corresponding trapdoor.

Next, note that A halts in step 3 with probability at most 2^{-n}. This is because the probability that $s'_t \in \left\{s_1^{(1)},\ldots,s_t^{(q(n))}\right\}$ is at most $\frac{q(n)}{2q(n)} = \frac{1}{2}$ for each $t \in [n]$ and therefore the probability this happens *for all* t is at most 2^{-n}. This also means that although the distribution of samples that A is invoked on is not precisely the same distribution on which A is guaranteed to operate, the two distributions are statistically close. Thus, with non-negligible probability, A outputs $x'_1,\ldots,x'_n$ such that $f_{\alpha_{s'_j,j}}(x'_j) = S(\alpha_{s'_j,j},r'_j)$ for all $j \in [n]$. In particular, $f_{\alpha_{s'_t,t}}(x'_t) = S(\alpha_{s'_t,t},r'_t)$. Since $\alpha = \alpha_{s'_t,t}$ and $r = r'_t$ we have that $f_\alpha(x'_t) = S(\alpha,r)$, i.e., x'_t is a preimage as required. □

3.4.3 Hardness of Enhanced TDP w.r.t Polynomially Many Samples

In this section we continue to establish connections between enhanced TDP that are hard to invert in the different scenarios introduced in Section 3.4.1. In this section we focus

on TDPs that are hard w.r.t *any* polynomial number of samples. This is in contrast to Section 3.4.2 in which we focused on hardness w.r.t a *fixed* number of samples.

The results presented in this section are depicted in Figure 3.4 using the same conventions as in Figure 3.3. We re-emphasize that throughout this section (and in particular in Figure 3.4) we consider TDP for which hardness (of inverting the permutation or of computing the hardcore predicate) holds w.r.t any (polynomial) number of samples.

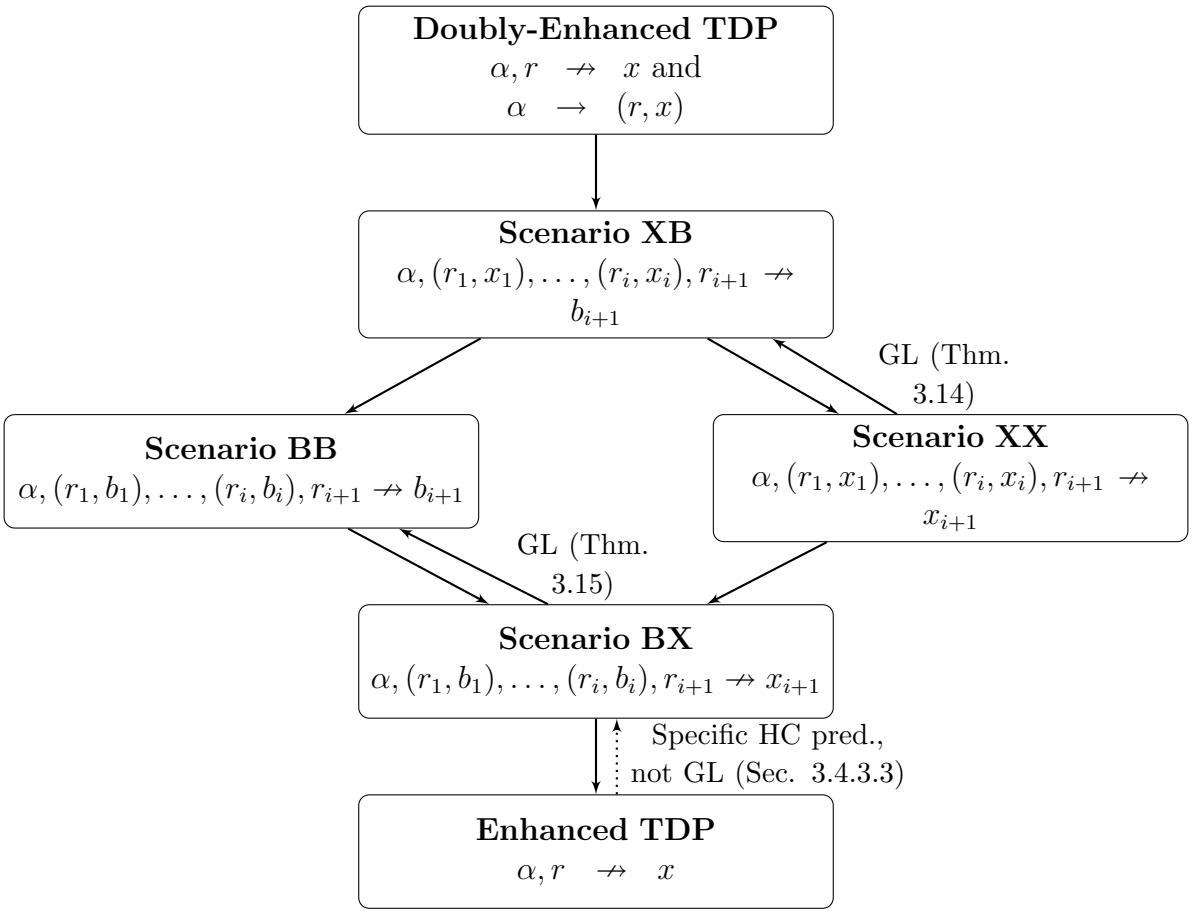

Figure 3.4: Hardness of Enhanced TDP w.r.t *any* (polynomial) number of samples.

3.4.3.1 Scenario XX vs. Scenario XB

Recall that in scenarios XX and XB, the adversary is required to invert the permutation or compute the hardcore predicate based on samples (r_j, x_j). An appealing aspect of Scenario XX is that it does not involve a hardcore predicate at all. In Appendix B, we show that a TDP that is hard to invert in the setting of Scenario XB suffices for the efficient prover non-interactive zero-knowledge protocol described in [Gol09]. In fact, we show that the only property of the TDP that is used in [Gol09] is hardness to invert in Scenario XB, which is seemingly a weaker assumption than the existence of doubly-enhanced TDP.

We proceed to show that the GL hardcore predicate of a TDP that is hard to invert in Scenario XX is unpredictable in Scenario XB. Thus, the GL hardcore predicate is hard to approximate even if the adversary is given polynomially many samples of the form (r_j, x_j).

Theorem 3.14. *Let $\{f_\alpha\}_\alpha$ be a TDP that is hard to invert in the setting Scenario XX. Then, the GL hardcore predicate, $h(x,s) = \langle x, s\rangle$, w.r.t the enhanced TDP $g_\alpha(x,s) = (f_\alpha(x), s)$ (where $|x| = |s|$), is hard to approximate in the setting of Scenario XB.*

Proof. The proof that h is an (enhanced) hardcore predicate of $\{g_\alpha\}$ (see [Gol01]), reduces the problem of inverting $\{g_\alpha\}$ to approximating h. We adapt the proof to show that inverting $\{g_\alpha\}$ given i samples of the form (r_j, x_j) reduces to approximating h given i' samples of the same form, where i' is related to i (via the advantage of the approximation).

We denote the domain sampling algorithm of $\{f_\alpha\}$ (resp. $\{g_\alpha\}$) by S_f (resp. S_g). We use the convention that random strings of S_f are denoted by r or r_j and those of S_g by (r,s) or (r_j, s_j) such that $S_g(\alpha); (r,s)) \stackrel{\text{def}}{=} (S_f(\alpha; r), s)$.

Let α be an index of a random permutation (w.r.t security parameter n), r a random string for S_f and x the corresponding preimage, i.e., $x = f_\alpha^{-1}(S_f(\alpha; r))$. We denote by $O(s)$ a machine that on input s returns $\langle x, s\rangle$ with probability $\frac{1}{2} + \epsilon$. The GL proof, describes an algorithm H that on input α, r and oracle access to O outputs x with probability $\frac{1}{\text{poly}(n, \frac{1}{\epsilon})}$. The number of oracle queries made by H is $q(n, \frac{1}{\epsilon})$, a polynomial in both n and $\frac{1}{\epsilon}$.

Assume toward a contradiction that there exists an adversary A that given $\alpha, (r_1, x_1), \ldots, (r_i, x_i), r_{i+1}$ computes $b_{i+1} = \langle x_{i+1}, s\rangle$ with advantage ϵ. We use A to construct an adversary A' that inverts $\{f_\alpha\}$ in the setting of Scenario XX using $i(n)$ blocks of $q(n, \frac{1}{\epsilon})$ samples. Thus, A' gets as input a permutation α, samples $(r_1, x_1), \ldots, (r_{i\cdot q(n,\frac{1}{\epsilon})}, x_{i\cdot q(n,\frac{1}{\epsilon})}$ and r' and find $x' = f_\alpha^{-1}(S(\alpha; r'))$. This is done by invoking $H(\alpha, r')$. The major issue is how to answer the oracle calls made by H. To answer the j-th oracle query, $O(s^{(j)})$, A' invokes A on α, the j-th block of samples that it is given as input and $(r', s^{(j)})$ and with probability $\frac{1}{2} + \frac{1}{\text{poly}(n, \frac{1}{\epsilon})}$ obtains $\langle x_{i+1}, s_{i+1}\rangle$. □

Note that the proof of Theorem 3.14 uses, in an essential way, the fact that $\{f_\alpha\}_\alpha$ is hard to invert given *any* polynomial number of samples because the number of samples is related to the advantage of the approximator.

3.4.3.2 Scenario BX vs. Scenario BB

The proof of Theorem 3.14 can be modified by replacing the (r_j, x_j) samples with (r_j, b_j) to prove the following theorem:

Theorem 3.15. *Let $\{f_\alpha\}_\alpha$ be a TDP that is hard to invert in the setting of Scenarion BX for any polynomial $i(\cdot)$. The GL hardcore predicate, $h(x,s) = \langle x, s\rangle$, w.r.t the enhanced TDP $g_\alpha(x,s) = (f_\alpha(x), s)$ (where $|x| = |s|$), is unpredictable in the setting of Scenario BB.*

3.4.3.3 Scenario BX

Theorem 3.9 constructs an enhanced TDP that is hard to invert in Scenario BX given any polynomial number of samples, based on any enhanced TDP. Therefore, it is relevant also in this section, when discussing scenarios in which the adversary is given polynomially many samples.

We stress that Theorem 3.9 constructs an enhanced TDP with a *specific* hardcore predicate (that is not the GL hardcore predicate). Therefore, Theorem 3.15 (that holds only for GL) *cannot* be applied to the constructed enhanced TDP to produce a TDP that is hard in Scenario BB.

Appendix A

An Enhanced TDP vulnerable in Scenario BB

In this section, we present a variant of the enhanced TDP suggested by Goldreich in [Gol04, Appendix C.1.]. Our TDP has the interesting property that it is completely vulnerable to an attack in Scenario BB. For this section we assume familiarity with basic number theory. Sufficient background is provided in [Gol01, Gol04].

Let N be a Blum integer, Q_N the set of quadratic residues modulo N and M_N the set of all integers in $\{1, \ldots, \lfloor \frac{N}{2} \rfloor\}$ with Jacobi symbol 1 modulo N. We define the predicate $QR_N \colon Z_N^* \to \{0,1\}$ to equal 1 if x is a quadratic residue modulo N and 0 otherwise.

Construction A.1. *(A factoring-based enhanced TDP)*

$I(1^n)$: *Uniformly at random select primes P and Q such that $2^{n-1} \leq P, Q \leq 2^n$ and set $N = PQ$. Select a random element $y \in_R M_N$. The index is (N, y) and the trapdoor is (P, Q).*

$S(N, y)$: *Select $r \in_R Z_N^*$. Set $z = y \cdot r^2 \bmod N$. If $z \leq \lfloor \frac{N}{2} \rfloor$ output z and otherwise output $N - z$.*

$F((N, y), x)$: *Set $z = x^2 \bmod N$. If $z \leq \lfloor \frac{N}{2} \rfloor$ output z and otherwise output $N - z$.*

$B((N, y), x)$: *Given the factorization of N it is possible to invert this permutation (for details see [Gol01, Gol04]).*

Note that Construction A.1 differs from one suggested in [Gol04] only in that the index includes an additional element y and the sampler that now multiplies by y. Indeed, as shown in [Gol04], F_N defines a permutation over M_N. The same argument can be applied to show that each element in M_N has exactly four preimages under $F_{N,y}$, therefore S samples uniformly from M_N.

We proceed by showing an *enhanced* hardcore predicate for the permutation. In particular this implies that this is an enhanced trapdoor permutation. Consider the predicate $h_{N,y}(x) = QR_N(F((N, y), x))$ (i.e., $h_{N,y}(x) = 1$ if and only if the image of x under $F_{N,y}$ is

a quadratic residue). Given x this predicate is easy to compute[1]. However, assuming the quadratic residuosity assumption, we show that this predicate is an *enhanced* hardcore predicate by showing that given $(N, y), r$, it is infeasible to approximate $QR_N(S(N, y; r))$.

The key observation is that multiplying by r^2 preserves the quadratic residuosity property whereas multiplying by $-r^2$ complements it (i.e., $y \cdot r^2$ is a quadratic residue if and only if y is a quadratic residue and $-y \cdot r^2$ is a residue if and only if y is a non-residue). Thus, given N, y and r it is easy to check whether y and $S(N, y; r)$ have the same QR_N value, i.e. compute $QR_N(y) \oplus QR_N(S(N, y; r))$, by checking whether S multiplies y by r^2 or by $-r^2$..

The above implies a reduction to the quadratic residuosity problem. Consider an adversary A that on input $(N, y), r$, computes $QR_N(S(N, y; r))$ with probability $\frac{1}{2} + \epsilon$. We use A to construct an adversary A' to the quadratic residuosity problem as follows. Given N and y, the adversary A' need to find $QR_N(y)$. This is done by selecting $r \in_R Z_N^*$, computing $b = QR_N(y) \oplus QR_N(S(N, y; r))$ (as described in the previous paragraph) and outputting $A((N, y), r) \oplus b$. With probability $\frac{1}{2} + \epsilon$ this equals $QR_N(S(N, y; r)) \oplus \Big(QR_N(y) \oplus QR_N(S(N, y; r)) \Big)$ which in turn equals $QR_N(y)$.

Thus, based on the quadratic residuosity assumption, Construction A.1 is an enhanced TDP. However, we argue that the enhanced hardcore bits are not pseudorandom. Indeed, the TDP is completely vulnerable to the attack as in Scenario BB. This follows from the fact that given N, r_1, r_2, it is easy to compute:

$$\Big(QR_N(y) \oplus QR_N(S(N, y; r_1)) \Big) \oplus \Big(QR_N(y) \oplus QR_N(S(N, y; r_2)) \Big)$$

which equals $QR_N(S(N, y; r_1)) \oplus QR_N(S(N, y; r_2))$.

[1] If $F((N, y), x) = x^2 \bmod N$, then $h_{N,y}(x) = 1$. Otherwise it must be that $F((N, y), x) = N - x^2 \bmod N$ which implies that $h_{N,y}(x) = 0$.

Appendix B

Non-Interactive Zero-Knowledge Proofs

Non-interactive zero-knowledge proofs are zero-knowledge proofs that consist of a single message sent from the prover to the verifier in the *common reference string* model. In this model, both the prover and verifier have access to a common reference string that is chosen uniformly at random by a trusted party. Feige et al. [FLS90] showed a non-interactive zero-knowledge proof for every language in $\mathcal{NP}$ based on the assumption that one-way functions exist. However, the prover of [FLS90] is inherently non-efficient, since it needs to invert the one-way function. As pointed out by Goldreich [Gol04, Gol09], the obvious solution of using trapdoor permutations fails and additional structure seems to be required. Thus, Goldreich [Gol09] introduced *doubly-enhanced* trapdoor permutations and showed that they suffice to construct such proofs with efficient provers. We mention that *efficient prover* non-interactive zero-knowledge proofs have proved to be extremely useful, especially for constructing encryption schemes that are secure against chosen ciphertexts attacks, as shown by Naor and Yung [NY90].

In this appendix we show that the additional structure that is actually used by [Gol09] is that the trapdoor permutation has a hardcore predicate that is hard to predict in the setting of Scenario XB, that is, given polynomially many samples of the form (r_j, x_j). Using Theorem 3.14 (that shows that the GL hardcore predicate of a trapdoor permutation, that is hard to invert in Scenario XX, is hard to predict in Scenario XB), we show how to construct efficient prover non-interactive zero-knowledge proofs for any $\mathcal{NP}$ language based on a trapdoor permutation that is hard to invert in Scenario XX, w.r.t polynomially many samples (seemingly a weaker assumption than doubly enhanced trapdoor permutations).

B.1 Efficient Prover Non-Interactive Zero-Knowledge Proofs

We start by defining non-interactive zero-knowledge proof systems with efficient provers.

Definition B.1. *A pair of probabilistic polynomial-time algorithms (P, V) are an* efficient prover non-interactive proof system *for an $\mathcal{NP}$ language L with the witness relation R_L if:*

- Completeness: *For every $z \in L$[1] and every witness w such that $(z, w) \in R_L$,*

$$\Pr_{r \in \{0,1\}^{poly(|z|)}}[V(z, r, P(z, w, r)) = 1] \geq \frac{2}{3}.$$

- Soundness: *For every $z \notin L$ and every algorithm P^*,*

$$\Pr_{r \in \{0,1\}^{poly(|z|)}}[V(z, r, P^*(z, r)) = 1] \leq \frac{1}{3}.$$

As usual, the error probability in both conditions can be made exponentially small by repetition (using independent random coins and reference strings). If completeness holds with probability 1, then the proof system is said to have perfect completeness.

Definition B.2. *An efficient prover non-interactive proof system (P, V) for a language L is* zero-knowledge *if there exists a probabilistic polynomial-time simulator M such that the ensembles:*

- $\{z, r, P(z, w, r)\}_{z \in L}$ *and*
- $\{M(z)\}_{z \in L}$

are computationally indistinguishable, where w is a witness for $z \in L$ (i.e., $(z, w) \in R_L$) and r is the common reference string (chosen uniformly at random).

B.2 Hidden Bits Model

As in [Gol09], we show a reduction to a restricted model in which the prover decides which bits on the reference string are available to the verifier. We remark that Feige et al. [FLS90] showed how to construct non-interactive zero-knowledge proofs for $\mathcal{NP}$ in this model.

Definition B.3. *A pair of probabilistic polynomial-time algorithms (P, V) are an* efficient prover non-interactive proof system in the hidden bits model *for an $\mathcal{NP}$ language L if:*

- Completeness: *For every $z \in L$ and every witness w such that $(z, w) \in R_L$*

$$\Pr_{r \in \{0,1\}^{poly(|z|)}}[V(z, r_J, (J, \pi)) = 1] \geq \frac{2}{3}$$

where $(J, \pi) \leftarrow P(z, w, r)$ and r_J is the substring of the common reference string r at positions $J \subseteq \{1, 2, \ldots, poly(|z|)\}$.

[1] We use the non-standard z since x is reserved to denote domain elements of a trapdoor permutation.

- Soundness: *For every $z \notin L$ and every algorithm P^*,*

$$\Pr_{r \in \{0,1\}^{poly(|z|)}}[V(z, r_J, (J, \pi)) = 1] \leq \frac{1}{3}$$

where $(J, \pi) \leftarrow P^*(z, r)$ and r_J is the substring of the common reference string r at positions $J \subseteq \{1, 2, \ldots, poly(|z|)\}$.

Zero-knowledge is defined as above, with the exception that we need to simulate $(z, r_J, P(z, w, r))$ (which equals (z, r_J, J, π)) rather than $(z, r, P(z, w, r))$.

B.3 Construction

Let L be a language and suppose (P, V) are an efficient prover non-interactive zero-knowledge proof system for L in the hidden-bits model, with soundness error 2^{-n-2}. We denote by $m(n)$ the length of the common reference string required by (P, V) for an input z of length n.

Our construction uses a TDP $\{f_\alpha : D_\alpha \to D_\alpha\}_\alpha$ with corresponding algorithms I, S, F, B and a hardcore predicate h that is non-uniformly hard to predict[2] in Scenario XB for polynomially many samples. We assume that on input 1^n, the index sampler I outputs an n-bit long index α and we denote by $\ell(n)$ the number of coins used by the domain sampler S when given such an index α of length n.

Construction B.4. *We construct a proof system (P', V') that for an input z of length n, uses a common reference of length $m(n) \cdot \ell(n)$ which is interpreted as $m = m(n)$ blocks of length $\ell = \ell(n)$ i.e., $r = (r_1, \ldots, r_m)$ where $|r_i| = \ell$.*

- Common Input: $z \in \{0,1\}^n$
- Prover's Auxiliary Input: w.
- Common Reference String: $r = (r_1, \ldots, r_m)$, *with* $r_i \in_R \{0,1\}^\ell$
- Prover (denoted P'):
  1. *Select $(\alpha, \tau) \leftarrow I(1^n)$ (where $n = |z|$).*
  2. *Using the trapdoor τ, compute $x_i = f_\alpha^{-1}(S(\alpha, r_i))$ and $b_i = h(x_i)$.*
  3. *Invoke P to obtain $(J, \pi) = P(z, w, b_1 \cdots b_m)$.*
  4. *Output $(\alpha, J, \pi, (x_1, \ldots, x_t))$, where $J = (j_1, \ldots, j_t)$.*
- Verifier (denoted V') *Given the proof $(\alpha, J, \pi, (x_1, \cdots, x_t))$:*
  1. *Verify that α is an index of a permutation, otherwise* halt and reject.

[2]That is, hard to predict even for a family of polynomial-sized circuits.

2. *Verify that* $S(\alpha, r_{j_i}) = f_\alpha(x_i)$*, for each* $j_i \in J$*. In case a mismatch is found* halt and reject.
3. *Compute* $b_i = h(x_i)$*, for* $i = 1, ..., t$.
4. *Invoke* V *on* $(x, b_1 \cdots b_t, J, \pi)$*, and* accept *if and only if* V *accepts.*

Note that in general, it is not clear that there is an efficient way to verify that α is an index of a permutation. Bellare and Yung [BY96] showed how to overcome this difficulty (by appending a proof that α is "almost a permutation"), however, they only considered the case $D_\alpha = \{0,1\}^{|\alpha|}$. In Section B.4 we discuss their solution and show that it extends to the general case.

Proposition B.5. *If* $\{f_\alpha\}_\alpha$ *is a TDP and* h *is a hardcore predicate of the collection, then Construction B.4 is an efficient prover non-interactive proof system.*

Proof. Let $z \in L$. We assume without loss of generality that h is completely unbiased (i.e., $\Pr[h(S(\alpha; r)) = 0] = \frac{1}{2}$)[3] and therefore the reference string seen by (P, V) is uniformly distributed. Thus, by completeness of (P, V), the prover P' accepts with probability $\geq \frac{2}{3}$.

To show that the proof is sound, assume that $z \notin L$. The key point is that each $\alpha \in \{0,1\}^n$, that the malicious prover might send, must be an index of a permutation or the verifier rejects. Thus, if we fix some $\alpha \in \{0,1\}^n$, the predicate h is completely unbiased and so, by soundness of (P, V), the verifier V' accepts with probability that is at most 2^{-n-2}. This holds for any *fixed* α, but if we take a union bound over all possible $\alpha \in \{0,1\}^n$, we obtain soundness error that is at most $\frac{1}{4}$. □

Proposition B.6. *If the hardcore predicate* h *is (non-uniformly) hard to predict in Scenario XB for polynomially many samples, then Construction B.4 is zero-knowledge.*

Proof. We start off with a simplifying assumption - that the number of bits revealed by P is a fixed function of m (instead of a distribution over $\{0, \ldots, m\}$). To justify this assumption, note that any prover in the standard version can be converted to this restricted version by doubling the length of the common reference string. The standard version prover is applied to the first half of the reference string and then we reveal a sufficient number of totally irrelevant bits of the second half.

To show that P' is zero-knowledge we convert any (efficient) simulator M for P into an (efficient) simulator M' for P'. The simulator M' operates as follows:

- Input: $z \in \{0,1\}^n$.
- Select $(\alpha, \tau) \leftarrow I(1^n)$.
- Compute $((\sigma_1, \ldots, \sigma_t), (j_1, \ldots, j_t), \pi) \leftarrow M(z)$.

[3] To justify this assumption, note that given a TDP $\{f_\alpha\}_\alpha$ with a hardcore predicate h, one can construct a TDP $\{g_\alpha\}_\alpha$ defined as $g(x\sigma) = f(x) \circ (\sigma \oplus h(x))$ whose last bit is a completely unbiased hardcore predicate.

- For each $i = 1, \ldots, t$, select at random a string $r_{j_i} \in \{0,1\}^\ell$ uniformly over all ℓ-bit strings r such that $\sigma_i = h(f_\alpha^{-1}(S(\alpha; r)))$. Note that this can be implemented using the trapdoor τ.
- For $i \in [m] \backslash \{j_k : \ k = 1, \ldots, t\}$, select $r_i \in_R \{0,1\}^\ell$.
- Output $(x, (r_1, \ldots, r_m), (\alpha, (j_1, \ldots, j_t), \pi, (x_1, \ldots, x_t)))$.

We proceed to show that the output of M' is computationally indistinguishable from the output of P'. The point is that the only difference between the simulation and the real view is that in the former the values on the common reference string do not necessarily match the values of the corresponding hidden bits seen by P (and that potentially appear in π). However, noticing this difference implies the ability to distinguish between a sequence of random bits and a sequence of hardcore bits. It is crucial to note that this distinguishing occurs in a context in which the distinguisher sees in addition random strings together with the preimages of the elements they sample. As discussed throughout this work, in this context, it may indeed by possible to distinguish between a sequence of random bits and a sequence of hardcore bits, when using a standard trapdoor permutation. This issue is addressed by using a TDP with a hardcore predicate that cannot be predicated even given this additional information, in other words, a hardcore predicate that cannot be predicting in the setting of Scenario XB. The straightforward argument follows.

Assume toward a contradiction that there exists a probabilistic polynomial-time algorithm A that distinguishes the output of $P'(z, w, r)$ from the output of $M'(z)$. Formally, this means that there exists a polynomial p and infinitely many $(z, w) \in R_L$:

$$|\Pr[A(P'(z, w, r)) = 1] - \Pr[A(M'(z)) = 1]| > \frac{1}{p(|z|)} \tag{B.1}$$

or actually without loss of generality[4]:

$$\Pr[A(P'(z, w, r)) = 1] - \Pr[A(M'(z)) = 1] > \frac{1}{p(|z|)}. \tag{B.2}$$

Fix (z, w) from the infinite set for which Eq. (B.1) holds. Based on the equivalence of unpredictability and pseudo-randomness, we use A to construct a probabilistic a family of polynomial-sized circuits B that have (z, w) hardwired and that given $\alpha, (r_1, x_1), \ldots, (r_m, x_m), r$ (where $x_i \stackrel{\text{def}}{=} f_\alpha^{-1}(S(\alpha; r_i))$) outputs $h(f_\alpha^{-1}(S(\alpha; r)))$. The circuit family B operates as follows:

- Input: $\alpha, (x_1, r_1), \ldots, (x_m, r_m), r$.

[4]Dropping the absolute value is justified by noting that if Eq. (B.1) holds for infinitely many $z \in L$ than there exists an infinite subset for which the distinguishing gap is greater than $\frac{1}{p(|z|)}$ or a infinite subset for which it is smaller than $-\frac{1}{p(|z|)}$. If the latter holds, we merely flip A's output).

- Hardwired: (z, w).
- Invoke $P(z, w, h(x_1), \ldots, h(x_m))$ to obtain (J, π). We denote by t the size of J (i.e. $t = |J|$) and by $\{j_1, \ldots, j_{m-t}\}$, the set $[m] \backslash J$.
- Select at random $k \in_R \{0, 1, 2, \ldots, m-t\}$
- For $j \in J$, $s_j = r_j$. These correspond to the revealed bits.
- For $j = j_1, \ldots, j_k$, set $s_j = r_j$. Corresponding to the k first hidden bits.
- Set $s_{j_{k+1}} = r$, i.e. the challenge random string.
- For $j = j_{k+2}, \ldots, j_{m-t}$, select $s_j \in_R \{0,1\}^\ell$.
- If $A(x, (s_1, \ldots, s_m), (\alpha, J, \pi, \{x_j\}_{j \in J})) = 1$, return $h(x_{j_{k+1}})$ otherwise, return the complement.

We denote by $A(\beta_1, \ldots, \beta_{m-t})$ the output of $A(x, (s_1, \ldots, s_m), (\alpha, J, \pi, \{x_j\}_{j \in J})) = 1$ conditioned on the event that the values of the hidden bits $(h(f_\alpha^{-1}(S(\alpha; s_j))$ are $\beta_1, \ldots, \beta_{m-t}$. Additionally, we denote $b_i \stackrel{\text{def}}{=} h(f_\alpha^{-1}(S(\alpha; s_{j_i})))$ and $\sigma_i \in_R \{0,1\}$. Note that using our notations and by the definition of B, it holds that $b_{k+1} = h(f_\alpha^{-1}(S(\alpha; s_{j_{k+1}}))) = h(f_\alpha^{-1}(S(\alpha; r)))$ which is the desired output of the algorithm. Using the definition of B and the fact that $\Pr_{\sigma \in_R \{0,1\}}[h(f_\alpha^{-1}(S(\alpha; r))) = \sigma] = \frac{1}{2}$ we have:

$$\begin{aligned}
\Pr[B \text{ succeeds}] &= \frac{1}{m-t+1} \sum_{k=0}^{m-t} \Big(\Pr[A(b_1, \ldots, b_k, \sigma_{k+1}, \ldots, \sigma_{m-t}) = 1 \wedge b_{k+1} = \sigma_{k+1}] \\
&\qquad + \Pr[A(b_1, \ldots, b_k, \sigma_{k+1}, \ldots, \sigma_{m-t}) = 0 \wedge b_{k+1} = \overline{\sigma_{k+1}}] \Big) \\
&= \frac{1}{2(m-t+1)} \sum_{k=0}^{m-t} \Big(\Pr[A(b_1, \ldots, b_k, b_{k+1}, \sigma_{k+2}, \ldots, \sigma_{m-t}) = 1] \\
&\qquad + 1 - \Pr[B(b_1, \ldots, b_k, \overline{b_{k+1}}, \sigma_{k+2}, \ldots, \sigma_{m-t}) = 1] \Big).
\end{aligned}$$

Now we note that:

$$\begin{aligned}
\Pr[A(b_1, \ldots, b_k, \sigma_{k+1}, \ldots, \sigma_{m-t}) = 1] = &\frac{1}{2} \Pr[A(b_1, \ldots, b_k, b_{k+1}, \sigma_{k+2}, \ldots, \sigma_{m-t}) = 1] \\
&+ \frac{1}{2} \Pr[A(b_1, \ldots, b_k, \overline{b_{k+1}}, \sigma_{k+2}, \ldots, \sigma_{m-t}) = 1]
\end{aligned}$$

and therefore the probability that B succeeds is:

$$\begin{aligned}
\frac{1}{2} + \frac{1}{m-t+1} \sum_{k=0}^{m-t} \Big(&\Pr[A(b_1, \ldots, b_k, b_{k+1}, \sigma_{k+2}, \ldots, \sigma_{m-t}) = 1] \\
&- \Pr[A(b_1, \ldots, b_k, \sigma_{k+1}, \ldots, \sigma_{m-t}) = 1] \Big)
\end{aligned}$$

which equals:

$$\frac{1}{2} + \frac{1}{m-t+1}\Big(\Pr[A(b_1,\ldots,b_{m-t})=1] - \Pr[A(\sigma_1,\ldots,\sigma_{m-t})=1]\Big).$$

Conditioned on the event that the hidden bits equal $b_1,\ldots,b_{m-t}$, the output of B is identically distributed to the output of A when given $P(z,w,r)$. On the other hand, conditioned on the hidden bits being totally random (i.e. $\sigma_1,\ldots,\sigma_{m-t}$), the output of B is *computationally indistinguishable* from the output of A when given $M'(z)$ (the distinction stems from the fact that B uses P to generate a "real" proof whereas M' uses M to generate a simulated proof). Thus, by Eq. (B.2), B computes the hardcore predicate in Scenario XB, with non-negligible advantage. □

B.4 Certifying Permutations

In Construction B.4 we explicitly assumed that it is possible to check whether an index α specifies a permutation. This assumption was used when proving that the protocol is sound. Note that if the prover provides an index that does not specify a permutation then the hardcore predicate might be heavily biased and used to break the soundness of the protocol. Alas, it seems as though certifying that an index is indeed a permutation is not always feasible. Bellare and Yung [BY96] addressed this issue by having the prover append to the proof, a non-interactive zero-knowledge proof that α is "almost a permutation", that is, that the fraction of domain elements that have more than one preimage via f_α is small. They showed that an "almost permutation" suffices to prove that soundness holds. Basically, their idea is for the prover to provide, as part of the proof, preimages of random elements specified by the common reference string. They proved that if the prover manages to provide sufficiently many preimages, then (with high probability) the index α describes a function that is almost a permutation, which in turn suffices to show that soundness of Construction B.4 holds.

Bellare and Yung considered permutations over $D_\alpha = \{0,1\}^{|\alpha|}$. In this case, it is easy to argue that the protocol is zero-knowledge, since the view of the verifier is easy to simulate by selecting random elements and computing their images. However, as is the case throughout this work, D_α is not necessarily $\{0,1\}^{|\alpha|}$ and is only required to be efficiently sampleable. The natural extension of the BY protocol to general D_α is to use the common reference string to specify sufficiently many *random strings* for the sampling algorithm and to give as a proof, the preimages of the corresponding sampled elements. Completeness and soundness hold as before. Zero-knowledge on the other hand follows from the fact that the permutation remains hard to invert (and the hardcore predicate hard to predict) even if the adversary is given random strings together with their corresponding preimages via f_α, that is, that $\{f_\alpha\}_\alpha$ is hard to invert in Scenario XX.

Bibliography

[Bar10] Boaz Barak. Cryptography course - Lecture notes, COS 433. Princeton University, Computer Science Department. Available at `http://www.cs.princeton.edu/courses/archive/spring10/cos433`, Spring 2010.

[Bla09] Eric Blais. Testing juntas nearly optimally. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 151–158. ACM, 2009.

[BM92] Mihir Bellare and Silvio Micali. How to sign given any trapdoor permutation. *JACM*, 39(1):214–233, 1992.

[BY96] Mihir Bellare and Moti Yung. Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *J. Cryptology*, 9(3):149–166, 1996.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *CACM: Communications of the ACM*, 28, 1985.

[ES81] Brad Efron and Charles Stein. The jackknife estimate of variance. *The Annals of Statistics*, 9(3):586–596, 1981.

[FLS90] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 308–317. IEEE Computer Society Press, 1990.

[Gam84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1984.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 169–178. ACM, 2009.

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-hop homomorphic encryption and rerandomizable yao circuits. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference*, pages 155–172. Springer, 2010.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32. ACM, 1989.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.

[Gol01] Oded Goldreich. *Foundations of Cryptography. Volume I: Basic Tools.* Cambridge University Press, 2001.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2: Basic Applications*. Cambridge University Press, 2004.

[Gol09] Oded Goldreich. Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: The state of the art. Available at `http://www.wisdom.weizmann.ac.il/~oded/PSBookFrag/nizk-tdp.ps`, November 2008 (revised October 2009).

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22th Annual ACM Symposium on Theory of Computing, STOC 1990*, pages 427–437, 1990.

[RAD78] Ronald L. Rivest, Leonard Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180. Academic Press, 1978.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.

[Yao82] A. C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23th Annual Symposium on Foundations of Computer Science (FOCS '82)*, pages 80–91, Los Alamitos, Ca., USA, November 1982. IEEE Computer Society Press.