Encryption Schemes from Bilinear Maps

Eu-Jin Goh

September 8, 2007

Dedication

This thesis is dedicated to my parents, Goh Ah Hock and Tan Chor Ngan, my aunt Tan Chor Koon, my sister Goh Yen-Li, and my wife Serene Koh. It is also dedicated to the memory of my late grandmother Tan Poh Tee.

Acknowledgements

This thesis would have been impossible without the unstinting support and mentoring of my advisor, Dan Boneh, and also the members of both the Applied Crypto and the Security Group at Stanford; in particular, Hovav Shacham, Nagendra Modadugu, Stanisław Jarecki, Philippe Golle, and John Mitchell.

Abstract

Encryption schemes are designed to provide data confidentiality and are a fundamental cryptographic primitive with many applications in higher-level protocols. Groups with a bilinear map allow us to build public key encryption schemes with new properties that are otherwise difficult to obtain using groups without a bilinear map. We support our thesis by presenting two encryption schemes based on bilinear groups; the first is a partial solution to the open problem on doubly homomorphic encryption proposed by Rivest et al. in 1978, and the second is the most efficient hierarchical identity based encryption scheme to date.

Our main result deals with homomorphic encryption. Using bilinear groups, we developed a homomorphic encryption scheme based on the subgroup decision complexity assumption; this encryption scheme is additively homomorphic and also possesses an additional limited (single) multiplicative homomorphism. Even with such limitations, our encryption scheme allows us to evaluate on encrypted inputs useful formulas such as polynomials of total degree at most two and dot products. Our encryption scheme also lends itself naturally to a secure function evaluation protocol for computing 2-DNFs, which can be used to improve private information retrieval protocols.

Our second result deals with hierarchical identity based encryption (HIBE), a generalization of identity based encryption. In previous constructions for HIBE, the length of ciphertexts, as well as the time needed for decryption, grows linearly with the depth of the hierarchy. Our HIBE system has ciphertext size, as well as decryption cost, that is independent of the hierarchy depth. The principal applications for HIBE are forward secure encryption and public key broadcast encryption. Using our HIBE system instead of existing HIBE systems in these two applications results in substantial reductions in the ciphertext size of both these applications.

Contents

Chapter 1

Introduction

Encryption schemes are designed to provide data confidentiality, and are a fundamental cryptographic primitive with many applications in higher-level protocols. Humans have been interested in encryption schemes since the Spartans who used simple transposition ciphers in 400 BC [1]. Modern encryption schemes have fortunately progressed greatly since the Spartans.

Modern encryption schemes are classified in two broad categories — 1) symmetric key encryption, and 2) public key encryption. In symmetric key encryption, the same secret key is used to both encrypt and decrypt; an example of a symmetric key cryptosystem is the block cipher AES [25]. In public key encryption, anyone can use the public key to encrypt data, but only the holder of the corresponding private key can decrypt ciphertexts; an example of a public key cryptosystem is the RSA encryption scheme [61]. By preventing the decryption of ciphertexts without knowledge of the secret key, encryption schemes thus provide data confidentiality. In this thesis, we focus on public key encryption.

Traditional public key encryption schemes, such as RSA and ElGamal [29] for example, are based on finite groups of prime or composite order. Our thesis deals with encryption schemes based on bilinear groups; bilinear groups are finite groups on certain elliptic curves that possess a bilinear map.

Our Thesis. Groups with a bilinear map allow us to build public key encryption schemes with new properties that are otherwise difficult to obtain using groups without a bilinear map.

Evidence. We support our thesis by presenting two encryption schemes based on bilinear groups; the first is a partial solution to the open problem on doubly homomorphic encryption proposed by Rivest et al. [60] in 1978, and the second is the most efficient hierarchical identity based encryption scheme to date.

Our main result deals with homomorphic encryption. An encryption scheme is additively (resp. multiplicatively) homomorphic if given the encryption of a message A and the encryption of B, anyone can compute the encryption of A + B (resp. A $\times$ B). An encryption scheme that is both additively and multiplicatively homomorphic is called doubly homomorphic. Known homomorphic encryption schemes that do not expand the ciphertext are either only additively or multiplicatively homomorphic, but not both. Developing a doubly homomorphic scheme is important because any logical function can be computed on ciphertext created using a doubly homomorphic encryption

scheme, which in turn gives an efficient solution to two party secure function evaluation in the honest but curious setting.

Using bilinear groups, we developed a encryption scheme based on the subgroup decision complexity assumption; this encryption scheme is additively homomorphic and also possesses an additional limited (single) multiplicative homomorphism. Even with such limitations, our encryption scheme allows us to evaluate on encrypted inputs useful formulas such as polynomials of total degree at most two and dot products. Our encryption scheme also lends itself naturally to a secure function evaluation protocol for computing 2-DNFs, which can be used to improve private information retrieval protocols. In fact, our schemes were used in 2006 to develop the first statistical non-interactive zero-knowledge argument for all NP languages [41], solving a 20 year old open problem.

Our second result deals with identity based encryption (IBE). An IBE system [65, 9] is a public key system where the public key can be an arbitrary string such as an email address. A central authority uses a master key to issue private keys to identities that request them. Hierarchical IBE (HIBE) [43, 34] is a natural extension of IBE that mirrors an organizational hierarchy. An identity at level k of the hierarchy tree can issue private keys to its descendant identities, but cannot decrypt messages intended for other identities.

In existing constructions for HIBE [43, 34, 5], the length of ciphertexts, as well as the time needed for decryption, grows linearly with the depth of the hierarchy. On the other hand, our HIBE system has ciphertext size, as well as decryption cost, that is independent of the hierarchy depth. The principal applications for HIBE are forward secure encryption [19] and public key broadcast encryption [54, 27]. Using our HIBE system instead of existing HIBE systems in these two applications results in substantial reductions in the ciphertext size of both these applications.

Previous Publications. The homomorphic encryption scheme of Chapter 3 based on the subgroup decision complexity assumption originally appeared in "Evaluating 2-DNF formulas on ciphertext," joint work with Dan Boneh and Kobbi Nissim, of which an extended abstract was presented at the Theory of Cryptography Conference in 2005 [12].

The hierarchical identity based encryption scheme of Chapter 4 originally appeared in "Hierarchical Identity Based Encryption with Constant Size Ciphertext," joint work with Dan Boneh and Xavier Boyen, of which an extended abstract was presented at Eurocrypt 2005 [7].

Chapter 2

Mathematical Background

We give a brief treatment of the bilinear groups underlying our encryption schemes, and also state the complexity assumptions used in our thesis. To gain more confidence in our complexity assumptions, we prove computational lower bounds in the generic group model on the difficulty of breaking a general assumption that encompasses two of our assumptions.

2.1 Bilinear groups

We use the following notation:

1. $\mathbb{G}$ and $\mathbb{G}_1$ are two (multiplicative) cyclic groups of finite order n.
2. g is a generator of $\mathbb{G}$.
3. e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$. In other words, for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$. We also require that $e(g, g)$ is a generator of $\mathbb{G}_1$.

We say that $\mathbb{G}$ is a bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and there exists both a group $\mathbb{G}_1$ and an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ as above.

2.2 Constructing bilinear groups of a given order n

One of our homomorphic constructions makes use of certain finite groups of composite order that support a bilinear map, and we now show how to construct such groups. Let $n > 3$ be a given square-free integer that is not divisible by 3. We construct a bilinear group $\mathbb{G}$ of order n as follows:

1. Find the smallest positive integer $\ell \in \mathbb{Z}$ such that $p = \ell n - 1$ is prime and $p = 2 \bmod 3$.

2. Consider the group of points on the (super-singular) elliptic curve $y^2 = x^3 + 1$ defined over $\mathbb{F}_p$. Since $p = 2 \bmod 3$ the curve has $p + 1 = \ell n$ points in $\mathbb{F}_p$. Therefore the group of points on the curve has a subgroup of order n which we denote by $\mathbb{G}$.

3. Let $\mathbb{G}_1$ be the subgroup of $\mathbb{F}^*_{p^2}$ of order n. The modified Weil pairing on the curve [49, 45, 10, 50] gives a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ with the required properties.

2.3 The subgroup decision problem

We define an algorithm $\mathcal{G}$ that given a security parameter $\tau \in \mathbb{Z}^+$ as input, outputs a tuple $(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e)$ where $\mathbb{G}, \mathbb{G}_1$ are groups of order $n = q_1 q_2$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map. On input τ, algorithm $\mathcal{G}$ works as follows:

1. Generate two random τ-bit primes q_1, q_2 and set $n = q_1 q_2 \in \mathbb{Z}$.
2. Generate a bilinear group $\mathbb{G}$ of order n as described at the end of Section 2.1. Let g be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be the bilinear map.
3. Output $(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e)$.

We note that the group action in $\mathbb{G}, \mathbb{G}_1$ as well as the bilinear map can be computed in polynomial time in τ.

Let $\tau \in \mathbb{Z}^+$ and let $(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e)$ be a tuple produced by $\mathcal{G}(\tau)$ where $n = q_1 q_2$. Consider the following problem: given $(n, \mathbb{G}, \mathbb{G}_1, e)$ and an element $x \in \mathbb{G}$, output '1' if the order of x is q_1 and output '0' otherwise; that is, without knowing the factorization of the group order n, decide if an element x is in a subgroup of $\mathbb{G}$. We refer to this problem as the *subgroup decision problem*. For an algorithm $\mathcal{A}$, we define the advantage of $\mathcal{A}$ in solving the subgroup decision problem as:

$$\text{SD-Adv}_{\mathcal{A}}(\tau) = \Big| \Pr\left[\mathcal{A}(n, \mathbb{G}, \mathbb{G}_1, e, x) = 1 \ : \ \begin{array}{c}(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e) \leftarrow \mathcal{G}(\tau), \\ n = q_1 q_2, \ \ x \xleftarrow{R} \mathbb{G}\end{array}\right] - \Pr\left[\mathcal{A}(n, \mathbb{G}, \mathbb{G}_1, e, x^{q_2}) = 1 \ : \ \begin{array}{c}(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e) \leftarrow \mathcal{G}(\tau), \\ n = q_1 q_2, \ \ x \xleftarrow{R} \mathbb{G}\end{array}\right] \Big|$$

where the probability is over the random choice of x in $\mathbb{G}$.

Definition 2.3.1. We say that $\mathcal{G}$ satisfies the subgroup decision assumption if for any polynomial time algorithm $\mathcal{A}$ we have that $\text{SD-Adv}_{\mathcal{A}}(\tau)$ is a negligible function in τ.

Informally, the assumption states that the uniform distribution on $\mathbb{G}$ is indistinguishable from the uniform distribution on a subgroup of $\mathbb{G}$. Recall that the factorization of the order of $\mathbb{G}$ is hidden so that the order of subgroups of $\mathbb{G}$ remains unknown to a polynomial time adversary.

2.4 The linear problem

We define an algorithm $\mathcal{G}$ that given a security parameter $\tau \in \mathbb{Z}^+$ as input, outputs a tuple $(p, \mathbb{G}, \mathbb{G}_1, e)$ where p is prime, $\mathbb{G}, \mathbb{G}_1$ are groups of order p and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map. On input τ, algorithm $\mathcal{G}$ works as follows:

1. Generate a random τ-bit prime p
2. Generate a bilinear group $\mathbb{G}$ of order p. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be the bilinear map.
3. Output $(p, \mathbb{G}, \mathbb{G}_1, e)$.

We note that the group action in $\mathbb{G}, \mathbb{G}_1$ as well as the bilinear map can be computed in polynomial time in τ.

Let $\tau \in \mathbb{Z}^+$ and let $(p, \mathbb{G}, \mathbb{G}_1, e)$ be a tuple produced by $\mathcal{G}(\tau)$. If g_1, g_2, and g_3 are arbitrary generators of $\mathbb{G}$, then the *linear problem* is defined as: given $g_1, g_2, g_3, g_1^a, g_2^b, g_3^c \in \mathbb{G}$ as input, output 1 if $a + b = c$ and 0 otherwise. The advantage of an algorithm $\mathcal{A}$ in deciding the linear problem is

$$\text{Linear-Adv}_{\mathcal{A}}(\tau) = \left| \Pr\left[\mathcal{A}(p, \mathbb{G}, \mathbb{G}_1, e, g_1, g_2, g_3, g_1^a, g_2^b, g_3^{a+b}) = 1 \ : \ \begin{array}{c}(p, \mathbb{G}, \mathbb{G}_1, e) \leftarrow \mathcal{G}(\tau), \\ g_1, g_2, g_3 \overset{\text{R}}{\leftarrow} \mathbb{G}, \ a, b \overset{\text{R}}{\leftarrow} \mathbb{Z}_p\end{array}\right] \right.$$
$$\left. - \Pr\left[\mathcal{A}(p, \mathbb{G}, \mathbb{G}_1, e, g_1, g_2, g_3, g_1^a, g_2^b, g_3^c) = 1 \ : \ \begin{array}{c}(p, \mathbb{G}, \mathbb{G}_1, e) \leftarrow \mathcal{G}(\tau), \\ g_1, g_2, g_3 \overset{\text{R}}{\leftarrow} \mathbb{G}, \ a, b, c \overset{\text{R}}{\leftarrow} \mathbb{Z}_p\end{array}\right] \right|$$

where the probability is over the random choices of g_1, g_2, g_3 in $\mathbb{G}$ and a, b, c in $\mathbb{Z}_p$.

Definition 2.4.1. We say that $\mathcal{G}$ satifies the linear decision assumption if for any polynomial time algorithm $\mathcal{A}$, we have that $\text{Linear-Adv}_{\mathcal{A}}(\tau)$ is a negligible function in τ.

The linear problem is well-defined in any group where DDH is also well-defined; it is usually used in bilinear groups.

2.5 Bilinear Diffie-Hellman Inversion Assumption

Let $\mathbb{G}$ be a bilinear group of order p. Let $w \in \mathbb{G}$ be a generator and $\beta \in \mathbb{Z}_p^*$. The ℓ-th Bilinear Diffie-Hellman Inversion problem [5, 51], denoted ℓ-BDHI, is as follows:

$$\ell\text{-BDHI} : \quad \text{given} \quad w, w^{\beta}, w^{(\beta^2)}, \ldots, w^{(\beta^\ell)} \quad \text{compute} \quad e(w, w)^{1/\beta} \tag{2.1}$$

Loosely speaking, the ℓ-BDHI assumption in $\mathbb{G}$ says that no efficient algorithm can solve ℓ-BDHI in $\mathbb{G}$ with non-negligible probability.

Weak BDHI Assumption. Our security theorems can be shown under a slightly weaker assumption, which we call Weak BDHI, and denote ℓ-wBDHI. This assumption is weaker in the sense that the ℓ-wBDHI assumption holds in any group where ℓ-BDHI holds, though the converse is not known to be true. To state the strongest results, we will use Weak BDHI throughout the paper.

Let g and h be two random generators of $\mathbb{G}$. Let α be a random number in $\mathbb{Z}_p^*$. We define the (equivalent) ℓ-wBDHI and ℓ-wBDHI* problems as follows:

$$\ell\text{-wBDHI} : \quad \text{given} \quad g, h, g^{\alpha}, g^{(\alpha^2)}, \ldots, g^{(\alpha^\ell)} \quad \text{compute} \quad e(g, h)^{1/\alpha} \tag{2.2}$$
$$\ell\text{-wBDHI*}: \quad \text{given} \quad g, h, g^{\alpha}, g^{(\alpha^2)}, \ldots, g^{(\alpha^\ell)} \quad \text{compute} \quad e(g, h)^{(\alpha^{\ell+1})} \tag{2.3}$$

These two problems are equivalent under a linear time reduction. The wBDHI problem (2.2) is seen to be more naturally related to the BDHI problem (2.1), but we will, for notational convenience, base our proofs on the wBDHI* problem (2.3).

It is easy to see that an algorithm for ℓ-wBDHI or ℓ-wBDHI* in $\mathbb{G}$ gives an algorithm for ℓ-BDHI with a tight reduction. Indeed, given a ℓ-BDHI problem instance $(w, w_1, \ldots, w_\ell)$, define the ℓ-wBDHI* instance $(w_\ell, h, w_{\ell-1}, \ldots, w_1, w)$ where $h = w_\ell^r$ for some random exponent $r \in \mathbb{Z}_p^*$. Let T' be the solution to this ℓ-wBDHI* problem instance, then $T = (T')^{1/r}$ is the solution to the original ℓ-BDHI instance.

We now define precisely the computational and decisional ℓ-wBDHI assumptions. For convenience, we define them in reference to the ℓ-wBDHI* problem. As shorthand, let $y_i = g^{(\alpha^i)} \in \mathbb{G}^*$. An algorithm $\mathcal{A}$ has advantage ϵ in solving the ℓ-wBDHI* problem in $\mathbb{G}$ if

$$\Pr\left[\mathcal{A}\big(g, h, y_1, \ldots, y_\ell\big) = e(g,h)^{(\alpha^{\ell+1})}\right] \geq \epsilon,$$

where the probability is over the random choice of generators g in $\mathbb{G}^*$, ,h in $\mathbb{G}$, the random choice of α in $\mathbb{Z}_p^*$, and the random bits used by $\mathcal{A}$. The decisional version of the ℓ-wBDHI* problem in $\mathbb{G}$ is defined in the usual manner, as follows. Let $\vec{y}_{g,\alpha,\ell} = (y_1, \ldots, y_\ell)$. An algorithm $\mathcal{B}$ that outputs $b \in \{0,1\}$ has advantage ϵ in solving Decision ℓ-wBDHI* in $\mathbb{G}$ if

$$\left|\Pr\left[\mathcal{B}\big(g, h, \vec{y}_{g,\alpha,\ell}, e(g,h)^{(\alpha^{\ell+1})}\big) = 0\right] - \Pr\left[\mathcal{B}\big(g, h, \vec{y}_{g,\alpha,\ell}, T\big) = 0\right]\right| \geq \epsilon,$$

where the probability is over the random choice of generators g in $\mathbb{G}^*$,h in $\mathbb{G}$, the random choice of α in $\mathbb{Z}_p^*$, the random choice of $T \in \mathbb{G}_1^*$, and the random bits consumed by $\mathcal{B}$. We refer to the distribution on the left as $\mathcal{P}_{wBDHI^*}$ and the distribution on the right as $\mathcal{R}_{wBDHI^*}$.

Asymptotically speaking, we say that $\mathbb{G}$ satisfies the wBDHI* assumption if any polynomial time algorithm's advantage in solving the ℓ-wBDHI* problem in $\mathbb{G}$ is negligible for any polynomial ℓ. For the rest of this paper, however, we will use the following concrete definition of the wBDHI assumption.

Definition 2.5.1. We say that the (Decision) (t, ϵ, ℓ)-wBDHI* assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the (Decision) ℓ-wBDHI* problem in $\mathbb{G}$.

For conciseness we occasionally drop the t and ϵ and simply refer to the (Decision) ℓ-wBDHI assumption in $\mathbb{G}$. As mentioned above, the computational and decisional ℓ-BDHI assumptions in $\mathbb{G}$ respectively imply the computational and decisional ℓ-wBDHI assumptions in $\mathbb{G}$. In Section 2.6.1 and 2.6.2, we show that a broad class of assumptions, including the ℓ-wBDHI assumption, hold in generic bilinear groups [66].

2.6 Diffie-Hellman Problems in Generic Bilinear Groups

A number of Diffie-Hellman-type complexity assumptions in bilinear groups have been used to construct efficient cryptosystems. These assumptions include the Bilinear DH assumption (BDH) [9], the Bilinear DH Inversion assumption (BDHI) [5], the Linear DH assumption [8], and the BDHE assumption used in [11], and others. To gain some confidence in these assumptions, including the BDHI assumption that is used in this thesis, one can prove computational lower bounds on the difficulty of breaking them in a generic bilinear group model [66]. Rather than prove a separate lower bound for each assumption, we give a general lower bound that encompasses all the assumptions listed above. This "master" generic-group lower bound can be used to qualify other assumptions that may come up in future constructions. We emphasize, however, that lower bounds in generic groups do not imply a lower bound in any specific group.

The Strong Diffie-Hellman (SDH) assumption [6] stands out from the list above and is not covered by the "master" argument. We briefly mention the reasons for this in Section 2.6.3.

2.6.1 General Diffie-Hellman Exponent Problem

Let p be an integer prime and let s, n be positive integers. Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$ and let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. Thus, P and Q are just two ordered sets containing s multi-variate polynomials each. We write $P = (p_1, p_2, \ldots, p_s)$ and $Q = (q_1, q_2, \ldots, q_s)$. We require that the first components of P, Q satisfy $p_1 = q_1 = 1$; that is, the constant polynomials 1. For a set Ω, a function $h : \mathbb{F}_p \to \Omega$, and a vector $x_1, \ldots, x_n \in \mathbb{F}_p$, we write

$$h(P(x_1, \ldots, x_n)) = \big(h(p_1(x_1, \ldots, x_n)),\ \ldots,\ h(p_s(x_1, \ldots, x_n)) \big) \in \Omega^s.$$

We use similar notation for the s-tuple Q. Let $\mathbb{G}_0, \mathbb{G}_1$ be groups of order p and let $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ be a non-degenerate bilinear map. Let $g \in \mathbb{G}_0$ be a generator of $\mathbb{G}_0$ and set $g_1 = e(g, g) \in \mathbb{G}_1$.

We define the (P, Q, f)-Diffie-Hellman Problem in $\mathbb{G}$ as follows: Given the vector

$$H(x_1, \ldots, x_n) = \big(g^{P(x_1,\ldots,x_n)},\ g_1^{Q(x_1,\ldots,x_n)} \big) \in \mathbb{G}_0^s \times \mathbb{G}_1^s,$$

compute $g_1^{f(x_1,\ldots,x_n)} \in \mathbb{G}_1$.

To obtain the most general result, we study the decisional version of this problem. We say that an algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the Decision (P, Q, f)-Diffie-Hellman problem in $\mathbb{G}_0$ if

$$\left| \Pr\Big[\mathcal{B}(H(x_1, \ldots, x_n),\ g_1^{f(x_1,\ldots,x_n)}) = 0\Big] - \Pr\Big[\mathcal{B}(H(x_1, \ldots, x_n),\ T) = 0\Big] \right| > \epsilon$$

where the probability is over the random choice of generator $g \in \mathbb{G}_0$, the random choice of $x_1, \ldots, x_n$ in $\mathbb{F}_p$, the random choice of $T \in \mathbb{G}_1$, and the random bits consumed by $\mathcal{B}$.

Before we can state the lower bound on the decision problem above, we need the following natural definition.

Definition 2.6.1. Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$. Write $P = (p_1, p_2, \ldots, p_s)$ and $Q = (q_1, q_2, \ldots, q_s)$ where $p_1 = q_1 = 1$. We say that a polynomial $f \in \mathbb{F}_p[X_1, \ldots, X_n]$ is dependent on the sets (P, Q) if there exist $s^2 + s$ constants $\{a_{i,j}\}_{i,j=1}^s,\ \{b_k\}_{k=1}^s$ such that

$$f = \sum_{i,j=1}^{s} a_{i,j} p_i p_j + \sum_{k=1}^{s} b_k q_k$$

We say that f is independent of (P, Q) if f is not dependent on (P, Q).

For a polynomial $f \in \mathbb{F}_p[X_1, \ldots, X_n]^s$, we let d_f denote the total degree of f. For a set $P \subseteq \mathbb{F}_p[X_1, \ldots, X_n]^s$ we let $d_P = \max\{d_f \mid f \in P\}$.

2.6.2 Complexity Lower Bound in Generic Bilinear Groups

We state the following lower bound in the framework of the generic group model [66]. We consider two random encodings ξ_0, ξ_1 of the additive group $\mathbb{Z}_p^+$, i.e. injective maps $\xi_0, \xi_1 : \mathbb{Z}_p^+ \to \{0, 1\}^m$. For $i = 0, 1$ we write $\mathbb{G}_i = \{\xi_i(x) \mid x \in \mathbb{Z}_p^+\}$. We are given oracles to compute the induced group action on $\mathbb{G}_0, \mathbb{G}_1$, and an oracle to compute a non-degenerate bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. We refer to $\mathbb{G}_0$ as a *generic* bilinear group. The following theorem gives a lower bound on the advantage of a generic algorithm in solving the decision (P, Q, f)-Diffie-Hellman problem. We emphasize, however, that a lower bound in generic groups does not imply a lower bound in any specific group.

Theorem 2.6.1. *Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$ and let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. Let $d = \max(2d_P, d_Q, d_f)$. Let ξ_0, ξ_1 and $\mathbb{G}_0, \mathbb{G}_1$ be defined as above. If f is independent of (P, Q) then for any $\mathcal{A}$ that makes a total of at most q queries to the oracles computing the group operation in $\mathbb{G}_0, \mathbb{G}_1$ and the bilinear pairing $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ we have:*

$$\left| \Pr\left[\mathcal{A}\left(\begin{array}{c} p,\ \xi_0(P(x_1,\ldots,x_n)), \\ \xi_1(Q(x_1,\ldots,x_n)), \\ \xi_1(t_0),\ \xi_1(t_1) \end{array} \right) = b \ : \ \begin{array}{c} x_1,\ldots,x_n, y \xleftarrow{\mathrm{R}} \mathbb{F}_p, \\ b \xleftarrow{\mathrm{R}} \{0,1\}, \\ t_b \leftarrow f(x_1,\ldots,x_n), \\ t_{1-b} \leftarrow y \end{array} \right] - \frac{1}{2} \right| \ \leq \ \frac{(q+2s+2)^2 \cdot d}{2p}$$

Proof. Consider an algorithm $\mathcal{B}$ that plays the following game with $\mathcal{A}$. Algorithm $\mathcal{B}$ maintains two lists of pairs,

$$L_0 = \{(p_i, \xi_{0,i}) : i = 1, \ldots, \tau_0\} \ , \qquad L_1 = \{(q_i, \xi_{1,i}) : i = 1, \ldots, \tau_1\} \ ,$$

under the invariant that at step τ in the game, $\tau_0 + \tau_1 = \tau + 2s + 2$. Here, $p_i \in \mathbb{F}_p[X_1, \ldots, X_n]$ and $q_i \in \mathbb{F}_p[X_1, \ldots, X_n, Y_0, Y_1]$ are multi-variate polynomials. The $\xi_{\star,\star}$ are strings in $\{0,1\}^m$.

The lists are initialized at step $\tau = 0$ by initializing $\tau_0 = s$ and $\tau_1 = s + 2$. We set $p_1, \ldots, p_s$ in L_0 to be the polynomials in P and we set $q_1, \ldots, q_s$ in L_1 to be the polynomials in Q. We also set $q_{s+1} = Y_0$ and $q_{s+2} = Y_1$. Algorithm $\mathcal{B}$ complete the preparation of the lists L_0, L_1 by setting the ξ-strings associated with distinct polynomials to random strings in $\{0,1\}^m$. Overall, initially L_0 contains s entries and L_1 contains $s + 2$ entries.

We can assume that $\mathcal{A}$ makes oracle queries only on strings obtained from $\mathcal{B}$, since $\mathcal{B}$ can make the strings in $\mathbb{G}_0, \mathbb{G}_1$ arbitrarily hard to guess by increasing m. We note that $\mathcal{B}$ can easily determine the index i of any given string $\xi_{j,i}$ in L_j (ties are broken arbitrarily). $\mathcal{B}$ starts the game by providing $\mathcal{A}$ with the value of p and a tuple of strings

$$\{\xi_{0,i}\}_{i=1}^{s}, \ \{\xi_{1,i}\}_{i=1}^{s+2},$$

meant to encode some tuple $\in \mathbb{G}_0^s \times \mathbb{G}_1^{s+2}$. Algorithm $\mathcal{B}$ responds to $\mathcal{A}$'s oracle queries as follows.

Group operation in $\mathbb{G}_0, \mathbb{G}_1$. A query in $\mathbb{G}_0$ consists of two operands $\xi_{0,i}, \xi_{0,j}$ with $1 \leq i, j \leq \tau_0$ and a selection bit indicating whether to multiply or divide the group elements. To answer, let $\tau_0' \leftarrow \tau_0 + 1$. Perform the polynomial addition or subtraction $p_{\tau_0'} \leftarrow p_i \pm p_j$ depending on whether multiplication or division is requested. If the result $p_{\tau_0'} = p_l$ for some $l \leq \tau_0$, then set $\xi_{0,\tau_0'} \leftarrow \xi_{0,l}$; otherwise, set $\xi_{0,\tau_0'}$ to a new random string in $\{0,1\}^m \setminus \{\xi_{0,1}, \ldots, \xi_{0,\tau_0}\}$. Insert the pair $(p_{\tau_0'}, \xi_{0,\tau_0'})$ into the list L_0 and update the counter $\tau_0 \leftarrow \tau_0'$. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with the string $\xi_{0,\tau_0'}$.

$\mathbb{G}_1$ queries are handled analogously, this time by working with the list L_1 and the counter τ_1.

Bilinear pairing. A query of this type consists of two operands $\xi_{0,i}, \xi_{0,j}$ with $1 \leq i, j \leq \tau_0$. To respond, $\mathcal{B}$ sets $\tau_1' \leftarrow \tau_1 + 1$, and performs the polynomial multiplication $r_{\tau_1'} \leftarrow p_i \cdot p_j$. If the result $q_{\tau_1'} = q_l$ for some $l \leq \tau_1$, it assigns $\xi_{1,\tau_1'} \leftarrow \xi_{1,l}$; otherwise, it sets $\xi_{1,\tau_1'}$ to a fresh random string in $\{0,1\}^m \setminus \{\xi_{1,1}, \ldots, \xi_{1,\tau_1-1}\}$. Finally, it adds $(q_{\tau_1'}, \xi_{1,\tau_1'})$ to the list L_1, updates $\tau_1 \leftarrow \tau_1'$, and outputs $\xi_{1,\tau_1'}$ as answer to the query.

After at most q queries, $\mathcal{A}$ terminates and returns a guess $b' \in \{0,1\}$. At this point $\mathcal{B}$ chooses random $x_1, \ldots, x_n, y \xleftarrow{\mathrm{R}} \mathbb{F}_p$ and $b \xleftarrow{\mathrm{R}} \{0,1\}$. Let $y_b = f(x_1, \ldots, x_n)$ and $y_{1-b} = y$.

For $i = 1, \ldots, n$, we set $X_i = x_i$, $Y_0 = y_0$, and $Y_1 = y_1$. It follows that the simulation provided by $\mathcal{B}$ is perfect unless the chosen random values for the variables $X_1, \ldots, X_n, Y_0, Y_1$ result in an equality relation between intermediate values that is not an equality of polynomials. In other words, the simulation is perfect unless for some i, j one of the following holds:

1. $p_i(x_1, \ldots, x_n) - p_j(x_1, \ldots, x_n) = 0$, yet the polynomials p_i and p_j are not equal.
2. $q_i(x_1, \ldots, x_n, y_0, y_1) - q_j(x_1, \ldots, x_n, y_0, y_1) = 0$, yet the polynomials q_i and q_j are not equal.

Let fail be the event that one of these two conditions holds. When event fail occurs, then $\mathcal{B}$'s responses to $\mathcal{A}$'s queries deviate from the real oracles' responses when the input tuple is derived from the vector $(x_1, \ldots, x_n, y_0, y_1) \in \mathbb{F}_p^{n+2}$.

We first bound the probability that event fail occurs. We bound the probability in two steps. First, consider setting $Y_b = f(X_1, \ldots, X_n)$. We claim that this symbolic substitution does not create any new equalities between polynomials q_i, q_j. In other words, if $q_i - q_j \neq 0$ for all i, j before the substitution, then $q_i - q_j \neq 0$ also holds after we set $Y_b = f(X_1, \ldots, X_n)$. This statement follows because f is independent of (P, Q). Indeed, $q_i - q_j$ is a polynomial of the form

$$\sum_{k=1}^{s}\sum_{l=1}^{s} a_{k,l}\, p_k\, p_l + \sum_{u=1}^{s} b_u\, q_u + c\, Y_0 + d\, Y_1$$

for some constants $a_{k,l}, b_u, c, d \in \mathbb{F}_p$. If this polynomial is non-zero but setting $Y_b = f(X_1, \ldots, X_n)$ causes this polynomial to vanish, then f must be dependent on (P, Q).

We are now left with polynomials in $X_1, \ldots, X_n, Y_{1-b}$. We need to bound the probability that for some i, j we get $(p_i - p_j)(x_1, \ldots, x_n) = 0$ even though $p_i - p_j \neq 0$, or that $(q_i - q_j)(x_1, \ldots, x, y) = 0$ even though $q_i - q_j \neq 0$. By construction, the maximum total degree of these polynomials is $d = \max(2d_P, d_Q, d_f)$. Therefore, for a given i, j the probability that a random assignment to $X_1, \ldots, X_n, Y_{1-b} \xleftarrow{\mathrm{R}} \mathbb{F}_p$ is a root of $q_i - q_j$ is at most d/p. The same holds for $p_i - p_j$. Since there are no more than $2\binom{q+2s+2}{2}$ such pairs (p_i, p_j) and (q_i, q_j) in total, we have that

$$\Pr[\mathsf{fail}] \leq \binom{q+2s+2}{2}\frac{2d}{p} \leq (q+2s+2)^2 d/p$$

If event fail does not occur, then $\mathcal{B}$'s simulation is perfect. Furthermore, in this case b is independent from algorithm $\mathcal{A}$'s view. Indeed b is only chosen after the simulation ends. Hence, $\Pr[b = b' | \neg\mathsf{fail}] = 1/2$. It now follows that

$$\begin{aligned} \Pr[b = b'] &\leq \Pr[b = b' | \neg\mathsf{fail}](1 - \Pr[\mathsf{fail}]) + \Pr[\mathsf{fail}] = \tfrac{1}{2} + \Pr[\mathsf{fail}]/2 \\ \Pr[b = b'] &\geq \Pr[b = b' | \neg\mathsf{fail}](1 - \Pr[\mathsf{fail}]) = \tfrac{1}{2} - \Pr[\mathsf{fail}]/2 \end{aligned}$$

and hence $|\Pr[b = b'] - \frac{1}{2}| \leq \Pr[\mathsf{fail}]/2 \leq (q+2s+2)^2 d/2p$ as required □

Corollary 2.6.2. Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$ and let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. Let $d = \max(2d_P, d_Q, d_f)$. If f is independent of (P, Q) then any $\mathcal{A}$ that has advantage $1/2$ in solving the decision (P, Q, f)-Diffie-Hellman Problem in a generic bilinear group $\mathbb{G}$ must take time at least $\Omega(\sqrt{p/d} - s)$.

Using Corollary 2.6.2. We briefly show that many standard (decisional) assumptions follow from Corollary 2.6.2.

- DDH in $\mathbb{G}_1$: set $P = (1), Q = (1, x, y), f = xy$.
- BDH in $\mathbb{G}_0$: set $P = (1, x, y, z), Q = (1), f = xyz$.
- ℓ-BDHI in $\mathbb{G}_0$: set $P = (1, x, x^2, \ldots, x^\ell), Q = (1), f = x^{2\ell+1}$.
- ℓ-BDHE in $\mathbb{G}_0$: set $P = (1, y, x, x^2, \ldots, x^{\ell-1}, x^{\ell+1}, \ldots, x^{2\ell}), Q = (1), f = x^\ell y$.

Extensions. To take advantage of certain families of elliptic curves, such as the so-called MNT curves [52], one often uses a more general bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_2$ where the groups $\mathbb{G}_0$ and $\mathbb{G}_1$ are not necessarily the same. To accurately model the algebraic structure of these groups, we let $\psi : \mathbb{G}_1 \to \mathbb{G}_0$ be an efficiently computable isomorphism (on MNT curves, such an isomorphism is given by the trace map). We briefly state a similar result for these more general maps. We first generalize the definition of independence.

Definition 2.6.2. Let $P, Q, R \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be three s-tuples of n-variate polynomials over $\mathbb{F}_p$. Write $P = (p_1, p_2, \ldots, p_s)$, $Q = (q_1, q_2, \ldots, q_s)$, and $R = (r_1, r_2, \ldots, r_s)$ where $p_1 = q_1 = r_1 = 1$. We say that a polynomial $f \in \mathbb{F}_p[X_1, \ldots, X_n]$ is dependent on the sets (P, Q, R) if there exist $2s^2 + s$ constants $\{a_{i,j}\}_{i,j=1}^s$, $\{b_{i,j}\}_{i,j=1}^s$, $\{c_k\}_{k=1}^s$ such that

$$f = \sum_{i=1}^{s}\sum_{j=1}^{s} a_{i,j}\, p_i\, q_j + \sum_{i=1}^{s}\sum_{j=1}^{s} b_{i,j} q_i\, q_j + \sum_{k=1}^{s} c_k\, r_k$$

We say that f is independent of (P, Q, R) if f is not dependent on (P, Q, R).

Given $P, Q, R \in \mathbb{F}_p[X_1, \ldots, X_n]^s$, and generators $g_0 \in \mathbb{G}_0,\ g_1 \in \mathbb{G}_1,\ g_2 \in \mathbb{G}_2$, we define the vector

$$H(x_1, \ldots, x_n) = \left(g_0^{P(x_1,\ldots,x_n)},\ g_1^{Q(x_1,\ldots,x_n)},\ g_2^{R(x_1,\ldots,x_n)} \right) \in \mathbb{G}_0^s \times \mathbb{G}_1^s \times \mathbb{G}_2^s$$

We say that an algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the Decision (P, Q, R, f)-Diffie-Hellman problem in $(\mathbb{G}_0, \mathbb{G}_1)$ if

$$\left| \Pr\left[\mathcal{B}(H(x_1, \ldots, x_n),\ g_1^{f(x_1,\ldots,x_n)}) = 0\right] - \Pr\left[\mathcal{B}(H(x_1, \ldots, x_n),\ T) = 0\right] \right| > \epsilon$$

where the probability is over the random choice of generators $g_0 \in \mathbb{G}_0,\ g_1 \in \mathbb{G}_1,\ g_2 \in \mathbb{G}_2$, the random choice of $x_1, \ldots, x_n$ in $\mathbb{F}_p$, the random choice of $T \in \mathbb{G}_2$, and the random bits consumed by $\mathcal{B}$. An identical proof to that of Theorem 2.6.1 gives the following theorem.

Theorem 2.6.3. *Let $P, Q, R \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be three s-tuples of n-variate polynomials over $\mathbb{F}_p$ and let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. Let $d = \max(d_P + d_Q, 2d_Q, d_R, d_f)$. If f is independent of (P, Q, R) then any $\mathcal{A}$ that has advantage $1/2$ in solving the decision (P, Q, R, f)-Diffie-Hellman Problem in a generic bilinear group $(\mathbb{G}_0, \mathbb{G}_1)$ must take time at least $\Omega(\sqrt{p/d} - s)$.*

2.6.3 The Case of the Strong Diffie-Hellman Assumption

As mentioned previously, the Strong Diffie-Hellman (SDH) assumption [6] is not covered by the General Diffie-Hellman Exponent framework. A slightly simplified version of the ℓ-SDH problem can be stated as follows:

> Given elements $g, g^{\alpha}, \ldots, g^{(\alpha^{\ell})}$ in $\mathbb{G}$, find a pair $(c, g^{1/(\alpha+c)})$ such that $c \neq 0 \bmod p$.

A closely related problem is to find $g^{1/(\alpha+c)}$ for some *given* value of $c \neq 0$. The two cases differ as follows:

- When c is fixed, the task is equivalent to the DHI problem [5, 28, 51], and is thus covered by the results of the previous section. We note that any hardness result on the *bilinear* DHI assumption immediately implies the hardness of the DHI assumption when the adversary no longer has access to the bilinear map oracle.
- When c is allowed to vary, as in the SDH problem [6], the General Diffie-Hellman Exponent framework no longer applies because the problem now admits a large number of valid, distinct solutions (one for each c). In this respect, the SDH assumption is fundamentally different from DHI. We note that Boneh and Boyen [6] give a direct proof of hardness of SDH in the generic group model.

Chapter 3

Homomorphic Encryption

Secure computation allows several parties to compute a function of their joint inputs without revealing more than what is implied by their own inputs and the function outcome. Any polynomial time functionality can be computed by a secure protocol that requires polynomial resources [70, 37]. These seminal results are obtained by a generic transformation that converts an insecure computation of a functionality to a secure version (often referred to as the 'garbled circuit' transformation).

Secure protocols generated from the garbled circuit transformation typically have poor efficiency. In particular, the communication complexity of the resulting protocols is proportional to the *size* of a circuit evaluating the functionality, and hence precludes sub-linear communication protocols. The result is that unless circuits are very small, the garbled circuit transformation is seldom used in protocols.

To avoid using the garbled circuit transformation, researchers have sought tools that give more efficient protocols for specific functionalities. *Homomorphic encryption* enables "computing with encrypted data" and is hence a useful tool for secure protocols. Current homomorphic public key systems [38, 29, 56] have limited homomorphic properties: given two ciphertexts $\mathsf{Encrypt}(\mathcal{PK}, x)$ and $\mathsf{Encrypt}(\mathcal{PK}, y)$, anyone can compute either the sum $\mathsf{Encrypt}(\mathcal{PK}, x + y)$, or the product $\mathsf{Encrypt}(\mathcal{PK}, xy)$, but not both.[1] The problem of constructing 'doubly homomorphic' encryption schemes where one may both 'add and multiply' is a long standing open question already mentioned by Rivest et al. [60].

Homomorphic encryption schemes have many applications, such as protocols for electronic voting schemes [22, 3, 23, 24], computational private information retrieval (PIR) schemes [46], and private matching [32]. Systems with more general homomorphisms (such as both addition and multiplication) will benefit all such problems.

Our contribution. We developed a homomorphic encryption scheme that is additively homomorphic and that is also capable of performing a single multiplication on encryption values. Our public key encryption scheme is based on finite groups of composite order that support a bilinear map. Using a construction along the lines of Paillier [56], we obtain a system with an additive homomorphism. In addition, the bilinear map allows for *one* multiplication on encrypted values. The security of this scheme is based on a new hardness assumption that we put forward – the

[1] An exception is the scheme by Sander et al. [63] that is doubly homomorphic over a semigroup. On the other hand, the homomorphism comes with the cost of a constant factor expansion per semigroup operation. See also its comparison with our results in Section 3 below.

subgroup decision problem. Namely, given an element of a group of composite order $n = q_1 q_2$, it is infeasible to decide whether it belongs to a subgroup of order q_1. As a result, our system supports arbitrary additions and one multiplication (followed by arbitrary additions) on encrypted data. This property in turn allows the evaluation of multi-variate polynomials of total degree 2 on encrypted values. Our applications follow from this new capability.

We also present the homomorphic public key encryption scheme based on the linear assumption introduced by Boneh, Boyen, and Shacham [8]. Boneh, Boyen, and Shacham observed that their linear encryption scheme was additively homomorphic. Based on a comment by Brent Waters, we show here that the bilinear map allows for *one* multiplication on encrypted values.

Applications. As a direct application of our new homomorphic encryption schemes, we can now evaluate quadratic multi-variate polynomials on ciphertexts provided the resulting value falls within a small set; in particular, we can compute dot products on ciphertexts. We use this property to create a gadget that enables the verification that an encrypted value is one of two 'good' values. We use this gadget to construct an efficient election protocol where voters do not need to provide proofs of vote validity. Finally, we generalize the election protocol to a protocol of universally verifiable computation.

In addition, we also construct a protocol for obliviously evaluating 2-DNFs. Our protocol gives a quadratic improvement in communication complexity over garbled circuits. We show how to get a private information retrieval scheme (PIR) as a variant of the 2-DNF protocol. Our PIR scheme is based on that of Kushilevitz-Ostrovsky [46] and improves the total communication in the basic step of their PIR protocol from $\sqrt{n}$ to $\sqrt[3]{n}$ for a database of size n.

In fact, using our homomorphic encryption schemes, Groth, Ostrovsky, and Sahai developed the first statistical non-interactive zero-knowledge argument for all NP languages [41], solving a long standing open problem in zero-knowledge. Our schemes were also used to create non-interactive Zaps for all NP languages [40].

Comparison to other public-key homomorphic systems. Most homomorphic systems provide only one homomorphism, either addition, multiplication, or xor. One exception is the system of Sander et al. [63] that provides the ability to evaluate NC^1 circuits on encrypted values. Clearly their construction also applies to 2-DNF formula. Unfortunately, the ciphertext length in their system grows exponentially in the depth of the 2-DNF formula when written using constant fan-in gates. In our system, the ciphertext size is independent of the formula size or depth; this property is essential for improving the communication complexity basic step of the Kushilevitz-Ostrovsky PIR protocol.

Furthermore, our schemes have the advantage of being able to directly evaluate on ciphertexts multi-variable quadratic polynomials of total degree two, unlike the Sanders et al. scheme where it is necessary to first convert the polynomial into a circuit.

Organization. The rest of this chapter is organized as follows. Section 3.1 details the construction of the two semantically secure public key encryption schemes, their security and homomorphic properties. The basic application to 2-DNF evaluation is presented in Section 3.2, followed by the election and universally verifiable computation protocols in sections 3.3 and 3.4. Section 3.5 summarizes our results and poses some open problems.

3.1 Two homomorphic public-key systems

We describe the subgroup decision scheme followed by the linear scheme.

3.1.1 Homomorphic system based on the subgroup decision problem

Our next system resembles the Paillier [56] and the Okamoto-Uchiyama [55] encryption schemes. We describe the three algorithms making up the system:

KeyGen(τ): Given a security parameter $\tau \in \mathbb{Z}^+$, run $\mathcal{G}(\tau)$ to obtain a tuple $(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e)$. Let $n = q_1 q_2$ where $q_1.q_2$ are two τ bit primes. Pick two random generators $g, u \stackrel{R}{\leftarrow} \mathbb{G}$ and set $h = u^{q_2} \in \mathbb{G}$. Then h is a random generator of the subgroup of $\mathbb{G}$ of order q_1. If $g = 1$ or $h = 1$, pick another two generators and repeat the steps until $g, h \neq 1$. The public key is $\mathcal{PK} = (n, \mathbb{G}, \mathbb{G}_1, e, g, h)$. The private key is $\mathcal{SK} = q_1$.

Encrypt($\mathcal{PK}, M$): We assume the message space consists of integers in the set $\{0, 1, \ldots, T\}$ with $T < q_2$. We encrypt bits in our main application, in which case $T = 1$. To encrypt a message m using public key $\mathcal{PK}$, pick a random $r \stackrel{R}{\leftarrow} [0, n-1]$ and compute

$$\mathrm{CT} = g^m h^r \in \mathbb{G}.$$

Output CT as the ciphertext.

Decrypt($\mathcal{SK}, \mathbf{CT}$): To decrypt a ciphertext CT using the private key $\mathcal{SK} = q_1$, observe that

$$(\mathrm{CT})^{q_1} = (g^m h^r)^{q_1} = (g^{q_1})^m \in \mathbb{G}$$

Let $\hat{g} = g^{q_1}$. To recover m, it suffices to compute the discrete log of $(\mathrm{CT})^{q_1}$ base $\hat{g}$. Since $0 \leq m \leq T$ this takes expected time $\tilde{O}(\sqrt{T})$ using Pollard's lambda method [48, p.128].

Note that decryption in this system takes polynomial time in the size of the message space T. Therefore, the system as described above can only be used to encrypt short messages. Clearly one can use the system to encrypt longer messages, such as session keys, using any mode of operation that converts a cipher on a short block into a cipher on an arbitrary long block. We note that one can speed-up decryption by precomputing a (polynomial-size) table of powers of $\hat{g}$ so that decryption can occur in constant time.

Homomorphic properties

The system is clearly additively homomorphic. Let $(n, \mathbb{G}, \mathbb{G}_1, e, g, h)$ be a public key. Given encryptions $\mathrm{CT}_1, \mathrm{CT}_2 \in G_1$ of messages $m_1, m_2 \in \{0, 1, \ldots, T\}$ respectively, anyone can create a uniformly distributed encryption of $m_1 + m_2 \bmod n$ by computing the product $\mathrm{CT} = (\mathrm{CT}_1)(\mathrm{CT}_2)h^r$ for a random $r \in [0, n-1]$.

More importantly, anyone can multiply two encrypted messages *once* using the bilinear map. Set $g_1 = e(g, g) \in \mathbb{G}_1$ and $h_1 = e(g, h) \in \mathbb{G}_1$. Then g_1 is of order n and h_1 is of order q_1. Also, write $h = g^{\alpha q_2} \in \mathbb{G}$ for some (unknown) $\alpha \in \mathbb{Z}$. Suppose we are given two ciphertexts $\mathrm{CT}_1 = g^{m_1} h^{r_1} \in \mathbb{G}$

and $\mathrm{CT}_2 = g^{m_2} h^{r_2} \in \mathbb{G}$. To build an encryption of the product $m_1 \cdot m_2 \bmod n$ given only CT_1 and CT_2, do: 1) pick a random $r \in [0, n-1]$, and 2) set $\mathrm{CT} = e(\mathrm{CT}_1, \mathrm{CT}_2) h_1^r \in \mathbb{G}_1$. Then

$$\begin{aligned} \mathrm{CT} = e(\mathrm{CT}_1, \mathrm{CT}_2) h_1^r = e(g^{m_1} h^{r_1}, g^{m_2} h^{r_2}) h_1^r &= g_1^{m_1 m_2} h_1^{m_1 r_2 + r_2 m_1 + \alpha q_2 r_1 r_2 + r} \\ &= g_1^{m_1 m_2} h_1^{\tilde{r}} \in \mathbb{G}_1 \end{aligned}$$

where $\tilde{r} = m_1 r_2 + r_2 m_1 + \alpha q_2 r_1 r_2 + r$ is distributed uniformly in $\mathbb{Z}_n^*$ as required. Thus, CT is a uniformly distributed encryption of $m_1 m_2 \bmod n$, but in the group $\mathbb{G}_1$ rather than $\mathbb{G}$ (this is why we allow for just one multiplication). Note that the system is still additively homomorphic in $\mathbb{G}_1$.

Note. In some applications we avoid blinding with h^r, making the homomorphic computation deterministic.

Quadratic polynomials. Let $F(x_1, \ldots, x_u)$ be a u-variate polynomial of total degree 2. The discussion above shows that given the encryptions $C_1, \ldots, C_u$ of values $x_1, \ldots, x_u$, anyone can compute the encryption of $C = F(x_1, \ldots, x_u)$. On the other hand, to decrypt C, the decryptor must already know that the result $F(x_1, \ldots, x_u)$ lies in a certain polynomial size interval.

Additional Algorithms

We now describe five additional useful algorithms that will be used in the 2-DNF protocols. We only describe the algorithms for ciphertexts belonging to $\mathbb{G}$ but the first two algorithms can be extended to $\mathbb{G}_1$ in a similar fashion.

$\mathsf{Rerandomize}(\mathcal{PK}, \mathbf{CT})$**:** Re-randomizes the input ciphertext. Given a ciphertext CT, pick a random $r \xleftarrow{\mathrm{R}} [0, n-1]$ and output $\mathrm{CT} \cdot h^r$.

$\mathsf{Blind}(\mathcal{PK}, \mathbf{CT}, r)$**:** Blinds the message encrypted in the ciphertext with message r. Given a ciphertext CT and a blinding factor $r \in [0, T]$, output $\mathsf{Rerandomize}(\mathcal{PK}, \mathrm{CT}^r)$.

$\mathsf{BlindBit}(\mathcal{PK}, \mathbf{CT})$**:** If the ciphertext contains an encryption of a bit b, creates a ciphertext containing $\neg b$. Given a ciphertext CT, output $\mathsf{Rerandomize}(\mathcal{PK}, g/\mathrm{CT})$.

$\mathsf{CreateWIProof}(\mathcal{PK}, b)$**:** Creates a witness indistinguishable zero-knowledge proof that a ciphertext contains an encryption of a bit. For a bit b, pick a random $r \xleftarrow{\mathrm{R}} [0, n-1]$ and output $(g^{2b-1} h^r)^r$ as the proof.

$\mathsf{VerifyWIProof}(\mathcal{PK}, \mathbf{CT}, \pi)$**:** Using the witness indistinguishable zero-knowledge proof π, verifies that the ciphertext contains an encryption of a bit. Given ciphertext CT and a witness indistinguishable proof π, check that $e(\mathrm{CT}, \mathrm{CT} \cdot g^{-1}) = e(h, \pi)$.

Creating and verifying the witness indistinguishable zero-knowledge proofs. The algorithms for creating and verifying the witness indistinguishable proofs are originally due to Groth, Ostrovsky, and Sahai [41], and were simplified by Boyen and Waters [16]. For a valid ciphertext, we see that $m \in \{0, 1\}$ if and only if either CT or $\mathrm{CT} \cdot g^{-1}$ is of order q_1. Thus, it is sufficient to prove that $e(\mathrm{CT}, \mathrm{CT} \cdot g^{-1})$ has order q_1. Also note that $e(\mathrm{CT}, \mathrm{CT} \cdot g^{-1}) = e(h, (g^{2b-1} h^r)^r)$ for $b \in \{0, 1\}$ where CT is an encryption of b. If this equation holds, then since h has order q_1, therefore $e(\mathrm{CT}, \mathrm{CT} \cdot g^{-1}) = e(h, (g^{2b-1} h^r)^r)$ has order q_1.

Security

We now turn to proving semantic security of the system under the subgroup decision assumption. Semantic security is the standard notion of security for encryption schemes and is typically defined using a game between an adversary and challenger. We first define semantic security before proving our scheme secure.

Semantic Security:

Setup: The challenger runs the $\mathsf{Keygen}(\tau)$ algorithm to create a public and private key, and sends the public key to the adversary.

Challenge: The adversary sends the challenger two equal length messages $M_0, M_1 \in \{0,1\}^*$. The challenger obtains a random bit $b \xleftarrow{R} \{0,1\}$, encrypts M_b with the public key, and sends the resulting ciphertext to the adversary.

Output: The adversary outputs a bit $b' \in \{0,1\}$ as its guess for b.

We define the advantage of such an adversary $\mathcal{B}$ in attacking the encryption scheme as $\left|\Pr[b = b'] - \frac{1}{2}\right|$. Roughly speaking, we say that an encryption scheme is semantically secure if no poly-time adversary has non-negligible advantage in winning the semantic security game.

Theorem 3.1.1. *The subgroup decision public key system is semantically secure assuming $\mathcal{G}$ satisfies the subgroup decision assumption.*

Proof. Suppose a polynomial time algorithm $\mathcal{B}$ breaks the semantic security of the system with non-negligible advantage $\epsilon(\tau)$. We construct an algorithm $\mathcal{A}$ that breaks the subgroup decision assumption with the same advantage. Given $(n, \mathbb{G}, \mathbb{G}_1, e, x)$ as input, algorithm $\mathcal{A}$ works as follows:

1. $\mathcal{A}$ picks a random generator $g \in \mathbb{G}$ and gives $\mathcal{B}$ the public key $(n, \mathbb{G}, \mathbb{G}_1, e, g, x)$.
2. Algorithm $\mathcal{B}$ outputs two messages $m_0, m_1 \in \{0, 1, \ldots, T\}$ to which $\mathcal{A}$ responds with the ciphertext $\mathrm{CT} = g^{m_b} x^r \in \mathbb{G}$ for a random $b \xleftarrow{R} \{0,1\}$ and random $r \xleftarrow{R} \{0, 1, \ldots, n-1\}$.
3. Algorithm $\mathcal{B}$ outputs its guess $b' \in \{0,1\}$ for b. If $b = b'$ algorithm $\mathcal{A}$ outputs 1 (meaning x is uniform in a subgroup of $\mathbb{G}$); otherwise $\mathcal{A}$ outputs 0 (meaning x is uniform in $\mathbb{G}$).

When x is uniform in $\mathbb{G}$, the challenge ciphertext CT is uniformly distributed in $\mathbb{G}$ and is independent of the bit b. Hence, in this case $\Pr[b = b'] = 1/2$. On the other hand, when x is uniform in the q_1-subgroup of $\mathbb{G}$, then the public key and challenge CT given to $\mathcal{B}$ are as in a real semantic security game. In this case, by the definition of $\mathcal{B}$, we know that $\Pr[b = b'] > 1/2 + \epsilon(\tau)$. It now follows that $\mathcal{A}$ satisfies $\text{SD-Adv}_{\mathcal{A}}(\tau) > \epsilon(\tau)$ and hence $\mathcal{A}$ breaks the subgroup decision assumption with advantage $\epsilon(\tau)$ as required. □

We note that if $\mathcal{G}$ satisfies the subgroup decision assumption then semantic security also holds for ciphertexts in $\mathbb{G}_1$. These ciphertexts are the output of the multiplicative homomorphism. If semantic security did not hold in $\mathbb{G}_1$, then it would also not hold in $\mathbb{G}$ because one can always translate a ciphertext in $\mathbb{G}$ to a ciphertext in $\mathbb{G}_1$ by "multiplying" by the encryption of 1. Hence, by Theorem 3.1.1, semantic security must also hold for ciphertexts in $\mathbb{G}_1$.

3.1.2 Homomorphic system based on the linear problem

We now present the homomorphic public key encryption scheme based on the linear assumption introduced by Boneh, Boyen, and Shacham [8]. They observed that their linear encryption scheme was additively homomorphic. We show later that the bilinear map allows for *one* multiplication on encrypted values. We describe the three algorithms making up the system:

$\mathsf{KeyGen}(\tau)$**:** Given a security parameter $\tau \in \mathbb{Z}^+$, run $\mathcal{G}(\tau)$ to obtain a tuple $(p, \mathbb{G}, \mathbb{G}_1, e)$. Choose a generator $g_3 \xleftarrow{R} \mathbb{G}$ and $u, w \xleftarrow{R} \mathbb{Z}_p$. Set $g_1 \leftarrow g_3^{1/u} \in \mathbb{G}$ and $g_2 \leftarrow g_3^{1/w} \in \mathbb{G}$. The public key is $\mathcal{PK} = (p, \mathbb{G}, \mathbb{G}_1, e, g_1, g_2, g_3) \in \mathbb{G}^3$. The private key is $\mathcal{SK} = (u, w) \in {\mathbb{Z}_p}^2$.

$\mathsf{Encrypt}(\mathcal{PK}, M)$**:** We assume the message space consists of integers in the set $\{0, 1, \ldots, T\}$ with $T < p$. To encrypt a message m using public key $\mathcal{PK}$, pick $r, s \xleftarrow{R} \mathbb{Z}_p$ and $(g_1^r,\ g_2^s,\ g_3^{-(r+s)+m}) \in \mathbb{G}^3$ as the ciphertext.

$\mathsf{Decrypt}(\mathcal{SK}, \mathbf{CT})$**:** To decrypt a ciphertext $\mathrm{CT} = (A, B, C)$ using the private key $\mathcal{SK} = (u, w)$, observe that for a properly formed ciphertext,

$$D = (A^u \cdot B^w \cdot C) = g_1^{ur} \cdot g_2^{ws} \cdot g_3^{-(r+s)+m} = g_3^m \in \mathbb{G}$$

To recover m, it suffices to compute the discrete log of D to the base g_3. Since $0 \le m \le T$ this takes expected time $\tilde{O}(\sqrt{T})$ using Pollard's lambda method [48, p.128].

As in the subgroup decision system, decryption in this system also takes polynomial time in the size of the message space T. Therefore, the system as described above can only be used to encrypt short messages. The techniques stated in the previous section for enciphering longer messages and for speeding up decryption can also be applied here.

Homomorphic properties

The system is clearly additively homomorphic. Let $\mathcal{PK} = (p, \mathbb{G}, \mathbb{G}_1, e, g_1, g_2, g_3)$ be a public key. Given encryptions $\mathrm{CT}_1 = (A_1, B_1, C_1), \mathrm{CT}_2 = (A_2, B_2, C_2) \in \mathbb{G}^3$ of messages $m_1, m_2 \in \{0, 1, \ldots, T\}$ respectively, anyone can create a uniformly distributed encryption of $m_1 + m_2 \bmod p$ by computing the product $\mathrm{CT} = (A_1 \cdot B_2 \cdot g_1^t,\ A_1 \cdot B_2 \cdot g_2^t, ,\ C_1 \cdot C_2 \cdot g_3^{-t}) \in \mathbb{G}^3$ for $t \xleftarrow{R} \mathbb{Z}_p$.

More importantly, anyone can multiply two encrypted messages *once* using the bilinear map. Suppose we are given two ciphertexts $\mathrm{CT}_1 = (A_1, B_1, C_1), \mathrm{CT}_2 = (A_2, B_2, C_2) \in \mathbb{G}^3$ that are encryptions of messages $m_1, m_2 \in \{0, 1, \ldots, T\}$. Let

$$h_1 = e(g_1, g_1),\ h_2 = e(g_1, g_2),\ h_3 = e(g_1, g_3),\ h_4 = e(g_2, g_2),\ h_5 = e(g_2, g_3),\ h_6 = e(g_3, g_3).$$

To build an encryption of the product $m_1 \cdot m_2 \bmod p$ given only C_1 and C_2, do: 1) pick a random $t \in \mathbb{Z}_p$, and 2) set the ciphertext as

$$\begin{aligned} \mathrm{CT} \;=\; \Big(& e(A, A') \cdot h_1^t,\ \ e(A, B') \cdot e(A', B) \cdot h_2^t,\ \ e(A, C') \cdot e(A', C) \cdot h_3^t, \\ & e(B, B') \cdot h_4^t,\ \ e(B, C') \cdot e(B', C) \cdot h_5^t,\ \ e(C, C') \cdot h_6^{-5t} \Big). \end{aligned}$$

For valid ciphertexts $\mathrm{CT}_1 = (g_1^{r_1}, g_2^{s_1}, g_3^{-(r_1+s_1)+m_1})$ and $\mathrm{CT}_2 = (g_1^{r_2}, g_2^{s_2}, g_3^{-(r_2+s_2)+m_2}) \in \mathbb{G}^3$, observe that

$$\begin{aligned}\mathrm{CT} \;=\; \Big(& h_1^{r_1 r_2 + t},\; h_2^{r_1 s_2 + r_2 s_1 + t},\; h_3^{r_1(-(r_2+s_2)+m_2) + r_2(-(r_1+s_1)+m_1) + t},\; h_4^{s_1 s_2 + t},\\ & h_5^{s_1(-(r_2+s_2)+m_2) + s_2(-(r_1+s_1)+m_1) + t},\; h_6^{(-(r_1+s_1)+m_1)(-(r_2+s_2)+m_2) - 5t}\Big). \end{aligned} \tag{3.1}$$

where the exponent of the h_6 term is distributed uniformly in $\mathbb{Z}_p$ as required. Thus, CT is a uniformly distributed encryption of $m_1 m_2 \bmod p$, but in the group $\mathbb{G}_1$ rather than $\mathbb{G}$ (this is why we allow for just one multiplication). Note that the system is still additively homomorphic in $\mathbb{G}_1$. In some applications we avoid blinding with h_1^r, h_2^r, h_3^r, h_4^r, h_5^r, h_6^r, making the homomorphic computation deterministic.

Note that the ciphertext for the linear system has 6 terms after a multiplication instead of 3. As a result, the decryption algorithm for the ciphertexts after a multiplication is slightly different.

$\mathsf{Decrypt}_{\mathbb{G}_1}(\mathcal{SK}, \mathbf{CT})$**:** To decrypt a ciphertext $\mathrm{CT} = (D, E, F, G, H, I) \in \mathbb{G}_1^6$ using the private key $\mathcal{SK} = (u, w)$, compute

$$J = D^{u^2} \cdot E^{uw} \cdot F^u \cdot G^{w^2} \cdot H^w \cdot I \in \mathbb{G}_1.$$

To recover the message, it suffices to compute the discrete log of J to the base h_6.

We now show that the decryption algorithm gives the correct answer for properly formed ciphertexts. For a properly formed ciphertext such as that in equation 3.1, we have

$$\begin{aligned} J \;&=\; \left(D^{u^2} \cdot E^{uw} \cdot G^{w^2}\right) \cdot (F^u \cdot H^w) \cdot I \\ &=\; \left(h_6^{(r_1+s_1)(r_2+s_2) + 3t}\right) \cdot \left(h_6^{(r_1+s_1)(-(r_2+s_2)+m_2) + (r_2+s_2)(-(r_1+s_1)+m_1) + 2t}\right) \\ &\qquad h_6^{(r_1+s_1)(r_2+s_2) - (r_1+s_1)(-(r_2+s_2)+m_2) - (r_2+s_2)(-(r_1+s_1)+m_1) + m_1 m_2 - 5t} \\ &=\; h_6^{m_1 m_2}. \end{aligned}$$

Quadratic polynomials. As in the subgroup decision system, the homomorphic properties of the subgroup decision system can also be used to compute quadratic polynomials.

Additional Algorithms

We now describe three additional useful algorithms that will be used in the 2-DNF protocols. We only describe the algorithms for ciphertexts belonging to $\mathbb{G}$ but the first two algorithms can be extended to $\mathbb{G}_1$ in a similar fashion.

$\mathsf{Rerandomize}(\mathcal{PK}, \mathbf{CT})$**:** Re-randomizes the input ciphertext. Given a ciphertext $\mathrm{CT} = (A, B, C)$, pick random $t, u \xleftarrow{\mathrm{R}} [0, n-1]$ and output $(A \cdot g_1^t, B \cdot g_2^u, C \cdot g_3^{-(t+u)})$.

$\mathsf{Blind}(\mathcal{PK}, \mathbf{CT}, r)$**:** Blinds the message encrypted in the ciphertext with r. Given a ciphertext $\mathrm{CT} = (A, B, C)$ and a blinding factor $r \in [0, T]$, compute $\mathrm{CT}' = (A, B, C \cdot g_3^r)$ and output $\mathsf{Rerandomize}(\mathcal{PK}, \mathrm{CT}')$.

$\mathsf{BlindBit}(\mathcal{PK}, \mathbf{CT})$**:** If the ciphertext contains an encryption of a bit b, creates a ciphertext containing $\neg b$. Given a ciphertext $\mathrm{CT} = (A, B, C)$, compute $\mathrm{CT}' = (1/A, 1/B, g_3/C)$ and then output $\mathsf{Rerandomize}(\mathcal{PK}, \mathrm{CT}')$.

Our linear system does not lend itself to creating non-interactive witness indistinguishable zero-knowledge proofs that a ciphertext is an encryption of 0 or 1. On the other hand, for interactive protocols, we can use standard techniques to create (less efficient) zero-knowledge proofs that a ciphertext created by our linear system is an encryption of a bit. For the rest of this paper, when the algorithms CreateWIProof and VerifyWIProof are invoked in our protocols for the linear system, we mean to say that we use general zero-knowledge techniques to create and verify these proofs interactively. We note that the variant of the linear system described by Groth, Ostrovsky, and Sahai [41] is amenable to creating non-interactive witness indistinguishable proofs of bit encryption; that variant can be substituted for our linear system in all our protocols.

Security

We now turn to proving semantic security of the system under the linear assumption.

Theorem 3.1.2. *The linear public key system is semantically secure assuming $\mathcal{G}$ satisfies the linear assumption.*

Proof. Suppose a polynomial time algorithm $\mathcal{B}$ breaks the semantic security of the system with non-negligible advantage $\epsilon(\tau)$. We construct an algorithm $\mathcal{A}$ that breaks the decision linear assumption with the same advantage. Given $\left(p, \mathbb{G}, \mathbb{G}_1, e, g_1, g_2, g_3, g_1^a, g_2^b, g_3^c\right) \in \mathbb{G}^6$ as input, algorithm $\mathcal{A}$ works as follows:

1. $\mathcal{A}$ gives algorithm $\mathcal{B}$ the public key (g_1, g_2, g_3).
2. Algorithm $\mathcal{B}$ outputs two messages $m_0, m_1 \in \{0, 1, \ldots, T\}$ to which $\mathcal{A}$ responds with the ciphertext $(g_1^{ar}, g_2^{br}, g_3^{-cr+m_b}) \in \mathbb{G}^3$ for $b \xleftarrow{\mathrm{R}} \{0,1\}$ and $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p$.
3. Algorithm $\mathcal{B}$ outputs its guess $b' \in \{0,1\}$ for b. If $b = b'$ algorithm $\mathcal{A}$ outputs 1 (meaning $c = a + b$); otherwise $\mathcal{A}$ outputs 0.

When $c \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, the challenge ciphertext C is uniformly distributed in $\mathbb{G}$ and is independent of the bit b. Hence, in this case $\Pr[b = b'] = 1/2$. On the other hand, when $c = a + b$, then the public key and challenge C given to $\mathcal{B}$ has the identical distribution as in a real semantic security game. In this case, by the definition of $\mathcal{B}$, we know that $\Pr[b = b'] > 1/2 + \epsilon(\tau)$. It now follows that $\mathcal{A}$ satisfies $\text{Linear-Adv}_{\mathcal{A}}(\tau) > \epsilon(\tau)$ and hence $\mathcal{A}$ breaks the decision linear assumption with non-negligible advantage $\epsilon(\tau)$ as required. □

We note that if $\mathcal{G}$ satisfies the decision linear assumption then semantic security also holds for ciphertexts in $\mathbb{G}_1$. These ciphertexts are the output of the multiplicative homomorphism. If semantic security did not hold in $\mathbb{G}_1$, then it would also not hold in $\mathbb{G}$ because one can always translate a ciphertext in $\mathbb{G}$ to a ciphertext in $\mathbb{G}_1$ by "multiplying" by the encryption of 1. Hence, by Theorem 3.1.2, semantic security must also hold for ciphertexts in $\mathbb{G}_1$.

3.1.3 Comparison

We briefly compare the subgroup decision and the linear schemes. The subgroup decision scheme creates ciphertexts that consist of only one group element whereas the linear scheme creates ciphertexts that consist of three group elements (six group elements after a homomorphic multiplication).

On the other hand, group elements in the subgroup scheme are from a composite order bilinear group whereas those in the linear scheme are from a prime order bilinear group (and are significantly smaller).

The arithmetic operations in the linear system are computed over prime order bilinear groups and will be faster than equivalent operations over the composite order bilinear groups used in the subgroup decision scheme. With this in mind, we see that encryption requires two exponentations in the subgroup scheme and three exponentiations in the linear scheme. Decryption requires one exponentiation in the subgroup scheme and two exponentiations (five exponentiations after a homomorphic multiplication) in the linear scheme, and both require a discrete log computation roughly of order square root the size of the message space.

With re-randomization, an homomorphic addition operation on two ciphertexts requires one exponentiation in the subgroup decision scheme, whereas it requires three exponentiations in the linear scheme. A homomorphic multiplication operation on two ciphertexts requires a pairing computation and one exponentiation in the subgroup decision scheme, whereas it requires nine pairings and three exponentiations in the linear scheme.

When used in the semi-honest 2-DNF protocol, the subgroup decision scheme, although simpler than the linear system, results in a more complicated protocol because of the need for Alice to validate the public key generated by Bob; this validation check is trivial in the linear system. On the other hand, the subgroup decision system gives very efficient non-interactive witness indistinguishable proofs of bit encryption, whereas we resort to using general interactive techniques to create and verify such proofs for the linear system. These proofs of bit encryption are used in the malicious Bob 2-DNF protocol. We note that Groth, Ostrovsky, and Sahai [41] present a variant of the linear system that is amenable to the construction of a non-interactive proof; their system can be used in this protocol for malicious Bob.

3.2 Two party efficient SFE for 2-DNF

We now show how to use our homomorphic encryption schemes to construct an efficient secure function evaluation protocol for 2-DNF. Our basic result is a direct application of the additive and multiplicative homomorphisms of our public key encryption schemes. We consider a two-party scenario where Alice holds a Boolean formula $\phi(x_1, \ldots, x_s)$ and Bob holds an assignment $a = a_1, \ldots, a_s$. As the outcome, Bob learns $\phi(a)$. We restrict our attention to 2-DNF formulas:

Definition 3.2.1. A 2-DNF formula over the variables $x_1, \ldots, x_s$ is given as $\vee_{i=1}^{k} (\ell_{i,1} \wedge \ell_{i,2})$ where $\ell_{i,1}, \ell_{i,2} \in \{x_1, \ldots, x_s, \bar{x}_1, \ldots, \bar{x}_s\}$.

We first give a protocol for the model of semi-honest parties, and then modify it to cope with a malicious Bob.

3.2.1 The basic protocol

In the semi-honest model, both parties are assumed to perform computations and send messages according to their prescribed actions in the protocol. They may also record whatever they see during the protocol (i.e. their own input and randomness, and the messages they receive). On the other hand, a malicious party may deviate arbitrarily from the protocol. We sketch the security definitions for the simple case where only one party (Bob) is allowed to learn the output. We refer readers to Goldreich's book [36] for the complete definitions.

Security in the semi-honest model. The definition is straightforward since only one party (Bob) is allowed to learn the output:

- Bob's security – indistinguishability: Alice cannot distinguish between the different possible inputs Bob may hold.
- Alice's security – Bob gets no information about ϕ beyond whether a satisfies ϕ.

Protocol `2-DNF` in Figure 3.1 efficiently evaluates 2-DNFs with semi-honest parties. Either the subgroup decision or the linear system can be used in the protocol. We get a three message protocol with communication complexity $O(s \cdot \tau)$ — a quadratic improvement in communication with respect to Yao's garbled-circuit [70] that yields communication proportional to the potential formula length, $\Theta(s^2)$. Recall that the message space of our two encryption schemes belongs to the interval $[0, T]$.

INPUT: Alice holds a 2-DNF formula $\phi(x_1, \ldots, x_s) = \vee_{i=1}^{k} (\ell_{i,1} \wedge \ell_{i,2})$ and Bob holds an assignment $a = a_1, \ldots, a_s \in \{0,1\}^s$. Both parties' inputs include a security parameter τ.

1. Bob performs the following:
   (a) He invokes $\mathsf{KeyGen}(\tau)$ to compute keys $\mathcal{SK}, \mathcal{PK}$, and sends $\mathcal{PK}$ to Alice.
   (b) He computes and sends $\mathsf{Encrypt}(\mathcal{PK}, a_j)$ for $j = 1, \ldots, s$.
2. Alice performs the following:
   (a) She computes an arithmetization Φ of ϕ by replacing "$\vee$" by "$+$", "$\wedge$" by "$\cdot$" and "$\bar{x}_j$" by "$(1 - x_j)$". Note that Φ is a polynomial in $x_1, \ldots, x_s$ with *total* degree 2.
   (b) Alice computes the encryption of $r \cdot \Phi(a)$ for a randomly chosen $r \in [0, T]$ using the encryption scheme's homomorphic properties and also the Blind algorithm. The result is sent to Bob.
3. If Bob receives an encryption of 0, he outputs 0; otherwise, he outputs 1.

Figure 3.1: Protocol `2-DNF`.

Claim 3.2.1. *Protocol* `2-DNF` *is correct for semi-honest Alice and Bob.*

Proof. For correctness, it suffices to show that Φ arithmetizes ϕ correctly when Φ is applied to Bob's assignment. When Φ is applied to Bob's assignment, each arithmetized $\wedge$ clause results in a 0 or 1 because Bob's assignment consists solely of n bits. Because Bob's assignment consists only of bits and also by comparing the definition of $\wedge$ against multiplication on bits, we see that the $\wedge$ clauses are computed correctly. Next, to compute the $\vee$ operations, the results of the $\wedge$ clauses (mulplication results) are summed up; if all of the $\wedge$ clauses are 0, then the summation is also 0; if n of the $\wedge$ clauses are 1, then the summation is n. By construction of the protocol, a summation result of 0 results in Bob outputting 0, and a summation result of any other value results in Bob outputting 1, which is the desired answer. □

Claim 3.2.2. *If the encryption scheme is semantically secure, then protocol* `2-DNF` *is secure against semi-honest Alice and Bob.*

Proof. Intuitively, Alice's security follows as the distribution on Bob's output only depends on whether ϕ is satisfied by a or not; Bob's security follows directly from the semantic security of the encryption scheme. We provide a simulation proof from Alice and Bob's view of the protocol.

We first consider the case when Alice is corrupt. Alice's view in an execution of `2-DNF` are the public key $\mathcal{PK}$ and n encryptions of bits generated using $\mathcal{PK}$. We now build a simulator that simulates Alice's view given access to only her inputs and output. The simulator uses Alice's security parameter τ as input to $\mathsf{KeyGen}(\tau)$ to compute keys $\mathcal{SK}, \mathcal{PK}$, and then writes $\mathcal{PK}$ on the transcript of Alice's view. Next, the simulator picks s random bits $a_1, \ldots, a_s \in \{0,1\}$ and writes the s ciphertexts generated by $\mathsf{Encrypt}(\mathcal{PK}, a_j)$ for $j = 1, \ldots, s$ onto the transcript. We now argue that the distribution of the simulated transcription is computationally indistinguishable from that of a real transcript. The same algorithm and input is used to generate the public key in both the real and simulated transcript; therefore $\mathcal{PK}$ has the same distribution in both transcripts. Because the encryption scheme is semantically secure, the distribution of an encryption of 0 is identical to that of an encryption of 1 using the same public key. As a result, the distribution of the n ciphertexts in the simulated transcript is also computationally indistinguishable from that of the real transcript, concluding the case when Alice is corrupt.

We next consider the case where Bob is corrupt. Bob's view in an execution of `2-DNF` is Alice's encrypted output of $r \cdot \Phi(a)$. We now build a simulator that simulates Bob's view given access to only his inputs and output. The simulator first picks $b \overset{\mathrm{R}}{\leftarrow} \{0,1\}$ and using Bob's public key $\mathcal{PK}$, the simulator computes $\mathrm{CT}_1 = \mathsf{Encrypt}(\mathcal{PK}, b)$ and $\mathrm{CT}_2 = \mathsf{Encrypt}(\mathcal{PK}, b)$. Next the simulator applies the multiplicative homomorphism to CT_1 and CT_2 to obtain CT_3, which is the encrypted message $b \cdot b$. Using the Blind algorithm, the simulator re-randomizes the message in CT_3 using r and then writes the result onto the transcript of Bob's view. We now argue that the distribution of the simulated transcription is identical to that of a real transcript. In the real transcript, if $\Phi(a) = 0$, then Bob receives a valid encryption of 0; if $\Phi(a) = 1$, then Bob receives a valid encryption of a random value $r \in [0, T]$. Furthermore, Bob receives only this encrypted result of the computation of ϕ applied to his assignment and nothing else. Furthermore, from Bob's view, the result from Alice is independent of his assignment because his only knowledge of ϕ is from this blinded encryption of a bit. Therefore, the distribution of the simulated transcript is computationally indistinguishable from that of the real transcript. □

Note. Protocol `2-DNF` (as well `Malicious-Bob-2-DNF` below) is secure even against a computationally unlimited Bob. Interestingly, the garbled circuit protocol (where Alice garbles ϕ) has the opposite property where it can be secured against an unbounded Alice but not an unbounded Bob. (See also Cachin et al. [17] for a discussion of computing on encrypted data versus garbled circuits).

3.2.2 Example application – private information retrieval

A private information retrieval (PIR) scheme [21, 46] allows a user to retrieve information from an s-bit database without revealing any information on which bit he is interested in. Symmetric private information retrieval (SPIR) is a PIR scheme that also protects the database privacy – a (semi-honest) user will learn only one of the database bits [35]. In this section, we show how an immediate application of protocol `2-DNF` results in a PIR/SPIR scheme. Our constructions can use either the linear or the subgroup decision encryption scheme and the protocol is based on that of Kushilevitz and Ostrovsky [46].

A SPIR scheme. We get a SPIR scheme with communication $O(\tau \cdot \sqrt{s})$ as an immediate application of protocol `2-DNF`. Without loss of generality, we assume that the database size s is a perfect square and treat the database as a table D of dimensions $\sqrt{s} \times \sqrt{s}$. Using this notation, suppose Bob wants to retrieve entry (I, J) of D. Alice (the database holder) holds the 2-DNF formula ϕ over $x_1, \ldots, x_{\sqrt{s}}, y_1, \ldots, y_{\sqrt{s}}$:

$$\phi(x_1, \ldots, x_{\sqrt{s}}, y_1, \ldots, y_{\sqrt{s}}) = \vee_{D_{i,j}=1} (x_i \wedge y_j),$$

and Bob's assignment a sets x_I and y_J to 1 and all other variables to 0. Bob and Alice carry out the `2-DNF` protocol with this assignment and 2-DNF formula. It is clear that $\phi(a) = D_{I,J}$ as required.

An alternative construction. Using the `2-DNF` protocol for SPIR restricts database entries to bits. We provide an alternative construction that allows each database entry to contain up to $O(\log n)$ bits. We consider the data as a table of dimensions $\sqrt{s} \times \sqrt{s}$ as above. To retrieve entry (I, J) of D, Bob creates two polynomials $p_1(x)$ and $p_2(x)$ of degree $\sqrt{s} - 1$ such that $p_1(i)$ is zero on $0 \leq i < \sqrt{s}$ except for $p_1(I) = 1$, and similarly $p_2(j)$ is zero on $0 \leq j < \sqrt{s}$ except for $p_2(J) = 1$. Bob sends to Alice the encryption of the coefficients of $p_1(x)$ and $p_2(x)$. Alice uses the encryption scheme's homomorphic properties to compute the encryption of

$$D_{I,J} = \sum_{0 \leq i,j < \sqrt{s}} p_1(i) p_2(j) D_{i,j}.$$

We allow $D_{i,j}$ to be b-bit values where $b = O(\log n)$. Bob recovers $D_{i,j}$ in time $O(2^{b/2})$ by computing a discrete logarithm e.g. using the baby-step giant-step algorithm.

A PIR scheme. Standard communication balancing of our SPIR scheme results in a PIR scheme where each party sends $O(\tau \cdot \sqrt[3]{s})$ bits. In particular, view the database as comprising of $s^{1/3}$ chunks, each chunk containing $s^{2/3}$ entries, where Bob is interested in retrieving entry (I, J, K) of D. Bob sends Alice the coefficients of two polynomials $p_1(x)$ and $p_2(x)$ of degree $\sqrt[3]{s} - 1$ such that $p_1(i) = p_2(i) = 0$ on $0 \leq i < \sqrt[3]{s}$ except for $p_1(I) = p_2(J) = 1$. Alice uses the encryption scheme's homomorphic properties to compute encryptions of

$$D_{I,J,k} = \sum_{0 \leq i,j < \sqrt[3]{s}} p_1(i) p_2(j) D_{i,j,k}$$

for $0 \leq k < \sqrt[3]{s}$. Alice sends the $\sqrt[3]{s}$ resulting ciphertexts to Bob who decrypts the Kth entry.

Recursively applying this balancing (as in Kushilevitz-Ostrovsky [46]) results in a protocol with communication complexity $O(\tau s^{\epsilon})$ for any $\epsilon > 0$. We note that the recursion depth to reach ϵ is lower in our case compared to that of Kushilevitz-Ostrovsky [46] by a constant factor of $\log_2 3$.

3.2.3 Security of the 2-DNF protocol against a malicious Bob.

The security definition for malicious parties captures both the privacy and correctness of the protocol and is limited to the case where only one of the parties is corrupt. Note that when considering malicious adversaries, actions such as refusing to participate in the protocol, aborting the protocol prematurely, and using different inputs from those given, cannot be prevented.

Informally, the security definition for malicious adversaries is based on a comparison with an ideal trusted party model (here the corrupt party may give an arbitrary input to the trusted

functionality). The security requirement is that for any strategy a corrupt party may play in a real execution of the protocol, there is an efficient strategy it could play in the ideal model with computationally indistinguishable outcomes. A complete definition of this ideal/real simulation paradigm can be found in the exposition of Yao's garbled circuits protocol in the malicious setting by Lindell and Pinkas [47].

In the basic 2-DNF protocol, a malicious Bob may try to learn about Alice's 2-DNF formula by sending Alice an encryption of a non-boolean assignment $a_1, \ldots, a_s$. He may also let Alice evaluate ϕ for an encrypted assignment that Bob cannot decrypt himself. Both types of behaviors do not correspond to a valid run in the ideal model. To prevent the first attack, we require that Bob creates witness indistinguishable zero-knowledge proofs for the encrypted assignments that allows Alice to ensure that a ciphertext she receives is an encryption of a bit. The second attack is prevented by Alice presenting Bob with a challenge that cannot be resolved unless he can decrypt; this decryption ability is used by the simulator for malicious Bob to create valid inputs for the trusted party.

Protocol `Malicious-Bob-2-DNF` described in Figure 3.2 evaluates 2-DNFs with malicious parties. As with the semi-honest protocol, either the subgroup decision or the linear encryption scheme can be used in the protocol. In this protocol, the subgroup decision scheme, although simpler than the linear system, results in a more complicated protocol because of the need for Alice to validate the public key generated by Bob; this validation check is trivial in the linear system. On the other hand, the subgroup decision system gives very efficient non-interactive witness indistinguishable proofs of bit encryption, whereas we resort to using general interactive techniques to create and verify such proofs for the linear system. Again we note that Groth, Ostrovsky, and Sahai [41] present a variant of the linear system that is amenable to the construction of a non-interactive proof; their system can be used in this protocol for malicious Bob.

Public Key Validation. In step 2 of the protocol, Bob has to prove to Alice that the public key is valid. For the linear encryption scheme, this step is trivial — for a public key $(p, \mathbb{G}, \mathbb{G}_1, e, g_1, g_2, g_3)$, Alice checks that $g_1, g_2, g_3 \in \mathcal{PK}$ are points in $\mathbb{G}$. For the subgroup decision encryption scheme, the procedure is more complicated and works as follows; for a public key $\mathcal{PK} = (n, \mathbb{G}, \mathbb{G}_1, e, g, h)$,

1. Bob proves to Alice in zero knowledge that n is a product of two primes [4, 68, 33, 18].
2. Alice checks that $g^n = h^n = 1 \in \mathbb{G}$ and that both $g \neq 1$ and $h \neq 1$. She rejects $\mathcal{PK}$ otherwise.

Claim 3.2.3. *If the encryption scheme used is semantically secure, then Protocol* `2-DNF` *is secure against semi-honest Alice and malicious Bob.*

Proof. In the case where Alice is semi-honest, the simulator that simulates Alice's view is identical to that for the (semi-honest) protocol `2-DNF` with three additions; our simulator here has to also simulate 1) the public key validation (step 2), 2) the proof of decryption ability (step 3), and 3) creating the witness indistinguishable zero-knowledge proofs (step 4). We first show how to simulate the public key validation step. For the linear system, there is nothing to simulate in public key validation step because there is no interaction between Alice and Bob. For the subgroup decision system, the simulator has to simulate the zero-knowledge interaction with Alice; this simulation can be done using standard techniques [4, 68, 33, 18]. For this step, the distribution of the simulated transcript is computationally indistinguishable from that in the real transcript by the security of the zero-knowledge protocol. We now show how to simulate the proof of decryption ability step. For

INPUT: as in protocol `2-DNF` in Figure 3.1.

1. Bob invokes $\mathsf{KeyGen}(\tau)$ to compute keys $\mathcal{SK}, \mathcal{PK}$ and sends $\mathcal{PK}$ to Alice.
2. Alice and Bob engage in a 'public key validation' protocol (see discussion below).
3. Next, Alice and Bob engage in a 'proof of decryption ability' protocol:
   (a) Alice chooses τ random bits $b_1, \ldots, b_\tau$ and then sends to Bob the encryptions $\mathsf{Encrypt}(\mathcal{PK}, b_1), \ldots, \mathsf{Encrypt}(\mathcal{PK}, b_\tau)$.
   (b) Bob replies with a decryption $b'_1, \ldots, b'_\tau$ of the received encryptions. Alice aborts if any of Bob's decryptions is incorrect.
4. Bob computes and sends $\mathrm{CT}_{a_j} \leftarrow \mathsf{Encrypt}(\mathcal{PK}, a_j)$ for $j = 1, \ldots, s$, as well as s corresponding witness indistinguishable zero-knowledge proofs (generated using $\mathsf{CreateWIProof}$) showing that each ciphertext CT_{a_j} is an encryption of a bit.
5. Alice performs the following:
   (a) She checks the witness indistinguishable zero-knowledge proofs using the algorithm $\mathsf{VerifyWIProof}$ to ensure that each CT_{a_j} is indeed an encryption of a bit. Alice aborts if any CT_{a_j} does not verify. Note that if the proofs are non-interactive, then Alice can verify the proofs interactively in Step 4.
   (b) She computes an arithmetization Φ of ϕ as in protocol `2-DNF`.
   (c) Using the homomorphic properties of the encryption scheme, she computes the encryption of $r \cdot \Phi(a)$ for randomly chosen $r \in [0, T]$. She sends the result to Bob.
6. If Bob receives an encryption of 0, he outputs 0; otherwise, he outputs 1.

Figure 3.2: Protocol `Malicious-Bob-2-DNF`.

both the linear or subgroup decision encryption scheme, the simulator uses the secret key generated in the first step of the simulation to run the $\mathsf{Decrypt}$ algorithm on Alice's challenge ciphertexts; the results of these decrypted messages are written to the transcript. For this step, the distribution of the simulated transcript is indistinguishable from that in a real transcript. Finally, in step 4, the simulator creates the proofs that the encrypted assignments are encryptions of bits by running the $\mathsf{CreateWIProof}$ algorithm on the assignment bits. Since the proofs are created according to the protocol, the distribution of the simulated and real transcripts are indistinguishable, which concludes the proof for a semi-honest Alice.

In the case of a malicious Bob, we build an ideal model simulator that has access to Bob and also to the trusted party, and can simulate the view of a real run of the protocol with Bob. The simulator interacts with malicious Bob and the trusted party as follows. The simulator runs the protocol and obtains the public key $\mathcal{PK}$ that Bob outputs in the first step. The simulator follows the protocol exactly (for either encryption scheme) in the public key validation step and also the proof of decryption ability step. In the proof of decryption ability step, the simulator remembers the τ ciphertexts $\mathrm{CT}_{b_1}, \ldots, \mathrm{CT}_{b_\tau}$ sent to Bob and their corresponding bits $b_1, \ldots, b_\tau$. When Bob outputs the encrypted assignments $\mathrm{CT}_{a_j} \leftarrow \mathsf{Encrypt}(\mathcal{PK}, a_j)$ for $j = 1, \ldots, s$ together with their corresponding witness indistinguishable proofs, the simulator checks the proofs to ensure that $\mathrm{CT}_{a_1}, \ldots, \mathrm{CT}_{a_s}$ are all encryptions of bits, and then does the following:

Procedure 1: Repeat the following steps s times; on the ith iteration

(a) The simulator rewinds the execution to the beginning of the proof of decryption ability step (step 3) of the protocol.

(b) The simulator chooses a random bit $b \xleftarrow{R} \{0,1\}$ and a random $k \in [1,\tau]$. If $b = 0$, the simulator runs $\mathsf{Rerandomize}(\mathcal{PK}, \mathrm{CT}_{a_i})$ to obtain a re-randomized CT_{a_i}; if $b = 1$, the simulator runs $\mathsf{BlindBit}(\mathcal{PK}, \mathrm{CT}_{a_i})$ to obtain a re-randomized and blinded CT_{a_i}.
Finally, the simulator sends Bob the encryptions CT_{b_j} for $j = 1, \ldots, \tau$ excluding k, as well as the re-randomized and possibly blinded CT_{a_i}; these τ ciphertexts are arranged in a uniformly random permutation and then sent to Bob.

(c) Bob replies with the decryption of the ciphertexts including the decryption of the re-randomized and possibly blinded ciphertext CT_{a_i}. The simulator checks that the messages $b_1, \ldots, b_{k-1}, b_{k+1}, \ldots, b_\tau$ are correctly decrypted and then unblinds CT_{a_i} to obtain the ith bit of Bob's assignment.

On every run, since 1) the encryption scheme is semantically secure, 2) CT_{a_i} is rerandomized and blinded, and 3) the ciphertexts are sent to Bob arranged according to a uniformly random permutation, a malicious Bob that wishes to lie about the assignment bit while still answering the $\tau-1$ challenge ciphertexts correctly can do no better than to guess at which is the re-randomized and blinded ciphertext and returning an incorrect value for that ciphertext. As a result, the probability of Bob successfully lying about the assignment bit on a single run is $1/\tau$. The probability of Bob successfully lying about at least one assignment bit over s runs is $1 - ((\tau-1)/\tau)^s$. To make the probability of Bob successfully lying about at least one assignment exponentially small, the simulator runs Procedure 1 $f(s)$ times where f is a polynomial function of s. We will later consider the event where Bob returns invalid encryptions causing the simulator to abort when arguing the computational indistinguishability of the execution between the ideal and real model.

The simulator then sends Bob's extracted assignment a to the trusted party and obtains $\phi(a)$ in return. The simulator rewinds the execution one final time to the beginning of the proof of decryption ability step (step 3) of the protocol, and sends the τ ciphertexts $\mathrm{CT}_{b_1}, \ldots, \mathrm{CT}_{b_\tau}$ to Bob. The rest of the execution proceeds as before and Bob eventually sends $\mathrm{CT}_{a_1}, \ldots, \mathrm{CT}_{a_s}$ again together with the corresponding witness indistinguishable proofs that that the ciphertexts are encryptions of bits. The simulator picks a random $r \in [0, T]$ ($T+1$ is the size of the message space of the encryption scheme), computes $r \cdot \phi(a)$, and sends the result of $\mathsf{Encrypt}(\mathcal{PK}, r \cdot \phi(a))$ to Bob. Finally, Bob outputs a bit as the result of the protocol.

We now argue that the view of malicious Bob in the ideal model is computationally indistinguishable from his view in the real model. Because the simulator follows the protocol exactly for the steps 1, 2, 4, and 6 in the protocol, the view of the Bob in those steps is computationally indistinguishable between the real and ideal model.

Next we argue that Bob's view in step 3 (proof of decryption ability) is computationally indistinguishable between the real and ideal model. We consider the first and last iteration of the step separately from the rewinding iterations. The first and the last iterations are identical and in these iterations, the simulator builds the challenge ciphertexts as specified in the protocol. Furthermore, the simulator aborts only if Bob returns incorrect decryptions of any of the ciphertexts; if this event occurred in an execution in the real model, Alice would also abort. Therefore, the distribution of the first and last iterations are computationally indistinguishable between the real and ideal model.

In each rewound iteration, except for the one encrypted assignment CT_{a_i}, the other challenge ciphertexts are created according to the protocol. Since the encrypted assignment has a corresponding witness indistinguishable zero-knowledge proof that it is a valid encryption of a bit, and also

because CT_{a_i} is re-randomized and blinded with a random bit, we see that CT_{a_i} is computationally indistinguishable from the other challenge ciphertexts assuming that the encryption scheme is semantically secure. Having established this fact, we now argue that if Bob did not send invalid decryptions of $\mathrm{CT}_{b_1}, \ldots, \mathrm{CT}_{b_\tau}$ in the first iteration (and hence causing the simulator to abort), Bob will not send invalid decryptions of $\mathrm{CT}_{b_1}, \ldots, \mathrm{CT}_{b_{k-1}}, \mathrm{CT}_{b_{k+1}}, \ldots, \mathrm{CT}_{b_\tau}$ in any rewound iteration (which causes the simulator to abort). Suppose Bob sends invalid decryptions of the challenge ciphertexts during a the ith rewound iteration. From Bob's view, the only difference between the first iteration (where Bob did not send invalid decryptions) and this failed iteration is that the simulator sent CT_{a_i} instead of CT_{b_k}. Using a standard hybrid argument, we can use Bob to break the semantic security of the encryption scheme. Since we assume that the encryption scheme is semantically secure, therefore Bob's view in step 3 (proof of decryption ability) is computationally indistinguishable between the real and ideal model.

We now consider Bob's view in step 5 of the protocol. By the argument after the description of Procedure 1, we see that the assignment extracted by the simulator from Bob's ciphertexts $\mathrm{CT}_{a_1}, \ldots, \mathrm{CT}_{a_s}$ obtained in the first iteration of steps 3 and 4 of the protocol is correct except with exponentially small probability. Furthermore, in the last iteration of the rewinding, the simulator feeds Bob the same challenge ciphertexts $\mathrm{CT}_{b_1}, \ldots, \mathrm{CT}_{b_\tau}$ in step 3, which results in Bob emitting the same set of encrypted assignments $\mathrm{CT}_{a_1}, \ldots, \mathrm{CT}_{a_s}$ in step 4. Because the simulator uses the correct assignment and follows the protocol exactly in this step, Bob's view of a non-aborting run of step 5 is computationally indistinguishable between the real and ideal model. We note that the ideal model simulator aborts at this step only if Bob had sent ciphertexts and proofs in the previous step that failed to verify; if this event occurred in an execution in the real model, Alice will also abort the protocol. Therefore, Bob's view of an aborting run of step 5 is also computationally indistinguishable between the real and ideal model.

We have shown that Bob's view of the protocol in the ideal model is computationally indistinguishable from his view in the real model, thus concluding our proof. □

3.3 An efficient election protocol without random oracles

In this section, we describe two electronic election protocols where voters submit boolean ("yes/no") votes. Such protocols were first considered by Benaloh and Fisher [22, 3] and more recently by Cramer et al. [23, 24].

A key component of electronic election schemes is a proof, attached to each vote, of its correctness (or validity); for example, a proof that the vote really is an encryption of 0 or 1. Otherwise, voters may corrupt the tally by sending an encryption of an arbitrary value. Such proofs of validity are typically zero-knowledge (or witness indistinguishable) proofs. These interactive zero knowledge proofs of bit encryption are efficiently constructed (using zero knowledge identification protocols) for standard homomorphic encryption schemes such as ElGamal [29, 44], Pedersen [57, 23], or Paillier [56, 26]. The proof of validity is then usually made non-interactive using the Fiat-Shamir heuristic of replacing communication with an access to a random oracle [31]. In the actual instantiation, the random oracle is replaced by some 'cryptographic function' such a hash function. Security is shown hence to hold in an ideal model with access to the random oracle, and not in the standard model [58].

Our first election protocol has the interesting feature that voters do not need to include proofs of validity or any other information except for their encrypted votes when casting their ballots.

Instead, the election authorities can jointly verify that a vote is valid based solely on its encryption. The technique is based on the following gadget:

A gadget for checking $\mathbf{c \in \{v_0, v_1\}}$. This gadget exploits our ability to evaluate a polynomial of total degree 2 on the encryption of c. We choose a polynomial that has v_0 and v_1 as zeros as follows: given an encryption of a value c, Alice uses the homomorphic properties of the encryption scheme to compute $r \cdot (c - v_0) \cdot (c - v_1)$ for a randomly chosen r. For $c \in \{v_0, v_1\}$, this computation results in the encryption of 0. For other values of c, the result is random. In the special case of $c \in \{0, 1\}$, Alice computes $r \cdot c \cdot (c - 1)$.

This gadget allows us to avoid using the Fiat-Shamir heuristic and yet makes our scheme efficient. As a result, our election scheme is very efficient from the voter's point of view as it requires only a single encryption operation (two exponentiations) to create a ballot. Curiously, this voting scheme is probably the most efficient for the voter, taking into account the efficiency of operating in an elliptic curve group. Either the subgroup decision or the linear encryption scheme can be used for this election protocol.

Our second election protocol is a variant of the first but instead of the election authorities using the gadget to verify the encrypted votes, each voter includes a witness indistinguishable proof that the ciphertext is an encryption of 0 or 1.

3.3.1 Election scheme 1

Our scheme belongs to the class of election protocols proposed by Cramer et al. [23, 24] where votes are encrypted using a homomorphic encryption scheme.

For robustness, we use threshold versions of the encryption schemes in Section 3.1. For simplicity (following Shoup [67]), we assume that a trusted dealer first generates the public/private keys, shares the private keys between the election authorities, and then deletes the private key (a generic secure computation may be used to replace the trusted dealer, as this is an offline phase). With this assumption, threshold versions of our encryption schemes can be constructed using standard techniques from discrete log threshold cryptosystems [57].

Correctness of threshold decryption. One caveat is that threshold decryption requires a zero knowledge of correct partial decryption from each election authority that contributes a share of its private key. Since the number of election authorities is typically a small constant, the proof of correct partial decryption can be performed interactively with relative efficiency between election authorities; transcripts of such interactions are made public for verification (note that transcripts do not leak information on votes). Another possible technique is to use a trusted source of random bits (such as a beacon [59]) among the election authorities, or for the authorities to collectively generate a public source of random bits. In a typical run of our protocol, the election authorities run only a limited number of these proofs (see below), hence the usage of either technique results in a reasonably efficient protocol, and allows us to avoid using the Fiat-Shamir heuristic.

We note that these techniques can also be used in existing election protocols for verifying a voter's ballot, which avoids the Fiat-Shamir heuristic; but the resulting protocol becomes unwieldy and inefficient especially when the number of voters is large (and we expect that there is at least several orders of magnitude more voters than election authorities).

Vote verification. Here we use the verification gadget of Section 3.3 in combination with threshold decryption. We let all authorities compute an encryption of $v \cdot (v - 1)$ and then jointly decrypt the

result. To save on computation, we check a batch of votes at once (i.e. $\sum r_i \cdot v_i \cdot (v_i - 1)$ where the r_i's are chosen by the verifiers) and then run a binary search to identify the invalid votes [2].

The protocol. We assume the existence of an online bulletin board where the parties participating in the protocol post messages. Our election protocol works as follows:

Setup: As discussed above, a trusted dealer first generates the public parameters and the private key for the either encryption scheme of Section 3.1, and shares the private key between the k election authorities so that at least t out the k election authorities are needed to decrypt. Finally, the trusted dealer deletes the private key and has no further role in the protocol. The public parameters are posted on a public bulletin board.

Denote one of the k election authorities as a leader (the election authority that organizes a quorum for decryption requests). After the public parameters are posted to the public board, the leader publishes an encryption of the bit 1 and the random bits used to create that encryption. Denote this encryption of the bit 1 as E_1. With the random bits, the other $k - 1$ election authorities can check that E_1 is indeed an encryption of 1.

Vote casting: Voters cast their ballots by encrypting a bit indicating their vote, and then publishing the encrypted bit to the public bulletin board.

Vote verification: When a ballot v has been posted, all k election authorities compute a ciphertext c corresponding to $v \cdot (v - 1)$ where E_1 is used as the encryption of "1" (hence, c is 'deterministic' given the encryption of v). The leader forms a quorum of $t - 1$ other election authorities to decrypt c (the other election authorities agree to participate only if c agrees with the ciphertext they computed). If c decrypts to something other than 0, then the vote v is invalid and is discarded.

For better efficiency in optimistic scenarios, any number of votes $v_1, \ldots, v_k$ can be verified in bulk by first computing $r_1 \cdot v_1 \cdot (v_1 - 1) + \ldots + r_k \cdot v_k \cdot (v_k - 1)$ where the r_i's are collectively chosen by the election authorities, and then checking that the decryption of the result is 0. All invalid votes are efficiently located by binary search. We note that in general it suffices for r_i to be relatively short, as the chance of $\sum r_i \cdot v_i \cdot (v_i - 1)$ being zero when some of the summed votes are invalid is exponentially small in $|r|$.

Vote tabulation and tally computation: After all the votes are posted and verified, all k election authorities each add all the valid encrypted votes on the public board together (using the additive homomorphic property of the encryption scheme) to form the tallied vote V. The leader obtains a quorum of election authorities to decrypt V. Each election authority decides whether to participate in the decryption request by comparing V with her own tally.

We note that our election protocol also possesses the necessary properties [22, 3, 23, 24] of voter privacy (from semantic security of the encryption scheme), universal verifiability (from the homomorphic property of the encryption scheme and also because all votes and proof transcripts are posted to the bulletin board), and robustness (from the threshold encryption scheme).

3.3.2 Election scheme 2

Unlike the first election scheme, this variant requires an encryption scheme that implements the CreateWIProof and VerifyWIProof algorithms non-interactively. We assume the existence of an

online bulletin board where the parties participating in the protocol post messages. Our election protocol works as follows:

Setup: Identical to election scheme 1 except that there is no need to create and publish the public encryption of 1 and its random bits.

Vote casting: Voters cast their ballots by encrypting a bit indicating their vote and also run the algorithm $\mathsf{CreateWIProof}$ to create a witness indistinguishable proof that their encrypted ballot is a encryption of either 0 or 1. Both the encrypted ballot and the proof is published on the bulletin board.

Vote verification: When a ballot has been posted, all k election authorities verify that the ballot is an encryption of 0 or 1 by running the $\mathsf{VerifyWIProof}$ algorithm on the corresponding proof. If the proof fails to verify for all the authorities, then the vote v is invalid and is discarded.

Vote tabulation and tally computation: Identical to election scheme 1.

3.4 Universally Verifiable Computation

We now describe a related application for the gadget of Section 3.3. Consider an authority performing a computation, defined by a (publicly known) circuit C over the joint private inputs $a = (a_1, \ldots, a_n)$ of the n users. The authority publishes the outcome $C(a)$ in a way that 1) lets everyone check that the computation was performed correctly, but 2) does not reveal any other information on the private inputs. Besides voting, other applications of universally verifiable computation include auctions.

To simplify our presentation, we only consider the case where $a_i \in \{0, 1\}$; general inputs are treated similarly using any binary representation. We describe a single authority protocol that is easily transformed into a threshold multi-authority protocol using standard methods.

3.4.1 A protocol for verifying $C(a)$

Setup. The authority uses $\mathsf{KeyGen}(\tau)$ to generate a public-key/private-key pair $\mathcal{PK}, \mathcal{SK}$. We assume the existence of a bulletin board where each user i posts an encryption $c_i = \mathsf{Encrypt}(\mathcal{PK}, a_i)$ of her input. We also assume the existence of a random function H accessible by all parties, which implies that we prove security only in the random oracle model. As in the previous section, we can do without a random oracle in the multi-authority case (details omitted).

We first give a high level overview of the protocol. After all users post their encrypted inputs onto the bulletin board, the authority decrypts each user's input and evaluates the circuit on these inputs. In the process of evaluating the circuit, the authority computes and publishes ciphertexts for all the wire values in C. In addition, the authority also publishes an encryption of the bit 1 and the random bits used to create that encryption. Denote this encryption of the bit 1 as E_1. Finally, the authority publishes an encrypted value V and a witness-indistinguishable proof that V is an encryption of 0; for now, we defer the exact definition of V.

To convince a verifier that the circuit was computed correctly, the authority needs to prove that 1) all inputs are binary, and 2) all gate outputs are correct. We show how a verifier checks that both conditions hold using validators v that can be publicly constructed from the public encrypted wire values. We first show how to construct validators for the user inputs and gate outputs before showing how to use these validators to verify the computation.

Building Validators for User Inputs. In the process of evaluating the circuit, the authority publishes the values on every wire of the circuit. We enumerate the wires and denote the value on wire i as a_i. We also denote the encryption of a_i as c_i. Recall that each user posts $c_i = \mathsf{Encrypt}(\mathcal{PK}, a_i)$ of her input a_i on the bulletin board (the a_i's are never revealed in the clear). For each input wire i with value a_i, let $r_i = H(i, c_i)$; note that r_i can be computed by any of the parties. The validator for a_i is $v_i = r_i \cdot a_i \cdot (1 - a_i)$, and the computation occurs modulo q_2 where q_2 is one of the factors of the modulus n (recall that $\mathcal{SK} = q_1$). It is easy to see that the encryption of v_i can be computed by anyone given H, c_i, and E_1.

We note that even given q_2 and allowing a polynomial (in the security parameter τ) number of applications of H, the probability that an adversary successfully generates an invalid c_i with $v_i = 0$ is bounded by $O(\text{poly}(\tau)/q_2)$, which is negligible in τ.

Building Validators for Gate Outputs. Let $g \in C$ be a binary gate for which both input wires x, y are validated. In addition, let $G(x, y)$ be the bivariate polynomial of total degree 2 that realizes the gate g. For example, an AND gate has $G_{\text{AND}}(x, y) = xy$, an OR gate has $G_{\text{OR}}(x, y) = x+y-xy$, and a NOT gate has $G_{\text{NOT}}(x) = 1 - x$. The validator for the output wire (enumerated z) of gate g is $v_z = r_z \cdot (a_z - G(x, y))$ where $r_z = H(z, c_z)$ and c_z is the encryption of the value a_z on wire z. Again, it is easy to see that any party can compute the encryption of v_z given H, c_x, c_y, c_z, and E_1.

Verifying the Circuit Using Validators. Using the homomorphic properties of the encryption scheme, anyone can compute (by herself) an encryption of the sum of validators for the circuit. Note that if all posted encryptions are correct, then the sum of validators is zero. Otherwise, it is zero with only a negligible probability. The authority supplies its own version of the encrypted validator sum called V, together with a zero-knowledge proof that the resulting sum is zero; in this case, the encryption is of the form h^r so to create the proof, one can either use protocols designed for the Pedersen encryption [57] or the $\mathsf{CreateWIProof}$ algorithm. To verify the circuit computation, a verifier computes her own validator sum V', checks that $V' = V$, and then verifies the witness-indistinguishable proof that V is an encryption of 0.

3.5 Summary and open problems

We presented a homomorphic encryption scheme that supports addition and one multiplication. We require that the values being encrypted lie in a small range as is the case when encrypting bits. These homomorphic properties enable us to evaluate multi-variate polynomials of total degree 2 given the encrypted inputs. We described a number of applications of the system. Most notably, using our encryption scheme, we (i) reduced the amount of communication in the basic step of the Kushilevitz-Ostrovsky PIR, (ii) improved the efficiency of election systems based on homomorphic encryption, and (iii) implemented universally verifiable secure computation. We hope this scheme will have many other applications. We end with a couple of open problems related to our encryption scheme:

n-linear maps. The multiplicative homomorphism was possible due to properties of bilinear maps. We note that an n-linear map would enable us to evaluate polynomials of total degree n rather than just quadratic polynomials. This provides yet another motivation for constructing cryptographic n-linear maps [15].

Message space. Our scheme is limited in the size of message space due to the need to compute

discrete logarithms during decryption. An encryption scheme that allows for a large message space would enable more applications, such as an efficient shared RSA key generation.

Chapter 4

Hierarchical Identity Based Encryption

An Identity Based Encryption (IBE) system [65, 9] is a public key system where the public key can be an arbitrary string such as an email address. A central authority uses a master key to issue private keys to identities that request them. Hierarchical IBE (HIBE) [43, 34] is a natural extension of IBE that mirrors an organizational hierarchy. An identity at level k of the hierarchy tree can issue private keys to its descendant identities, but cannot decrypt messages intended for other identities (details are given in Section 4.1.1). The first construction for HIBE is due to Gentry and Silverberg [34] where security is based on the Bilinear Diffie-Hellman (BDH) assumption in the random oracle model. A subsequent construction due to Boneh and Boyen [5] gives an efficient (selective-ID secure) HIBE based on BDH without random oracles. In both constructions, the length of ciphertexts and private keys, as well as the time needed for decryption and encryption, grows linearly in the depth ℓ of the hierarchy.

There are currently two principal applications for HIBE. The first, due to Canetti, Halevi, and Katz [19], is forward secure encryption. Forward secure encryption enables users to periodically update their private keys so that a message encrypted at period n cannot be read using a private key from period $n' > n$. To provide for $T = 2^t$ time periods, the CHK construction uses a HIBE of depth t where identities are *binary* vectors of length at most t. At time n, the encryptor encrypts using the identity corresponding to the n-th node of this depth t binary tree. Consequently, using previous HIBE systems [34, 5], ciphertexts in this forward secure construction are of size $O(t)$; private keys are of size $O(t^2)$ but can be reduced to size $O(t)$ by using updateable public storage. The second application for HIBE, due to Dodis and Fazio [27], is using HIBE to convert the NNL broadcast encryption system [54] into a *public-key* broadcast system. Unfortunately, the resulting public-key broadcast system is no better than simpler constructions because ciphertext length in previous HIBE constructions is linear in the depth of the hierarchy.

Our Contribution. We present a HIBE system where the ciphertext size as well as the decryption cost are *independent* of the hierarchy depth ℓ. Ciphertexts in our HIBE system are always just three group elements and decryption requires only two bilinear map computations. Private keys in our basic system contain ℓ group elements as in previous HIBE constructions.

Our system gives a forward secure encryption system with short ciphertexts consisting of only three group elements, for any number $T = 2^t$ of time periods. With our basic HIBE system, the

private key size in this forward secure encryption system is $O(t^2)$. In Section 4.2 we describe a hybrid system that borrows some features from the Boneh-Boyen HIBE [5] and results in a forward secure encryption scheme where private key size is reduced to $O(t^{3/2})$ and ciphertext size is $O(\sqrt{t})$. By using updateable public storage as in CHK [19], private key size in these systems can be further reduced to size $O(t)$ and $O(\sqrt{t})$ respectively. In addition, instantiating the Dodis-Fazio [27] system with our HIBE system results in a *public-key* broadcast system that is as efficient as the NNL subset difference method.

It is worth noting that private keys in our system *shrink* as the identity depth increases; this shrinkage is the opposite behavior from previous HIBE systems where private keys become larger as we descend deeper down the hierarchy tree. This behavior leads to "limited delegation" where an identity at depth k can be given a restricted private key that only lets it issue private keys to descendants of limited depth (as opposed to any descendant).

Security of our system is based on the previously used Bilinear Diffie-Hellman Inversion assumption [5, 28, 51]. This assumption was described in Section 2.5. In Section 4.1 we describe our HIBE system and prove its security in the selective identity model without using random oracles. We then observe that a selective-ID secure HIBE results in a fully secure HIBE in the random oracle model. In Sections 4.2 and 4.3 we discuss several extensions and applications of the system. For example, in addition to the applications already mentioned, we show how private keys can be further compressed to sublinear size and also describe an efficient mechanism for encrypting to the future.

4.1 A HIBE System with Constant Size Ciphertext

Let $\mathbb{G}$ be a bilinear group of prime order p and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be a bilinear map. For now, we assume that public keys (that is, identities ID) at depth k are vectors of elements in $(\mathbb{Z}_p^*)^k$. We write $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$. The j-th component corresponds to the identity at level j. We later extend the construction to public keys over $\{0,1\}^*$ by first hashing each component I_j using a collision resistant hash $H : \{0,1\}^* \to \mathbb{Z}_p^*$. We also assume that the messages to be encrypted are elements in $\mathbb{G}_1$. The HIBE system works as follows:

$\mathtt{Setup}(\ell)$**:** To generate system parameters for an HIBE of maximum depth ℓ, select a random generator $g \in \mathbb{G}$, a random $\alpha \in \mathbb{Z}_p^*$, and set $g_1 = g^\alpha$. Next, pick random elements $g_2, g_3, h_1, \ldots, h_\ell \in \mathbb{G}$. The public parameters and the master key are

$$\mathit{params} = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell) \,, \qquad \mathit{master\text{-}key} = g_2^\alpha.$$

$\mathtt{KeyGen}(d_{\mathsf{ID}|k-1}, \mathsf{ID})$**:** To generate a private key d_{ID} for an identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$ of depth $k \le \ell$, using the master secret, pick a random $r \in \mathbb{Z}_p^*$ and output

$$d_{\mathsf{ID}} = \left(g_2^\alpha \cdot \left(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3\right)^r, \quad g^r, \quad h_{k+1}^r, \quad \ldots, \quad h_\ell^r \right) \in \mathbb{G}^{2+\ell-k}.$$

Note that d_{ID} becomes shorter as the depth of ID increases.

The private key for ID can be generated incrementally, given a private key for the parent identity $\mathsf{ID}_{|k-1} = (\mathrm{I}_1, \ldots, \mathrm{I}_{k-1}) \in (\mathbb{Z}_p^*)^{k-1}$, as required. Indeed, let

$$d_{\mathsf{ID}|k-1} = \left(g_2^\alpha \cdot \left(h_1^{\mathrm{I}_1} \cdots h_{k-1}^{\mathrm{I}_{k-1}} \cdot g_3\right)^{r'}, \quad g^{r'}, \quad h_k^{r'}, \ldots, h_\ell^{r'} \right) = (a_0, a_1, b_k, \ldots, b_\ell)$$

be the private key for $\mathsf{ID}_{|k-1}$. To generate d_{ID}, pick a random $t \in \mathbb{Z}_p^*$ and output

$$d_{\mathsf{ID}} = \left(a_0 \cdot b_k^{\mathrm{I}_k} \cdot \left(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3\right)^t, \quad a_1 \cdot g^t, \quad b_{k+1} \cdot h_{k+1}^t, \quad \ldots, \quad b_\ell \cdot h_\ell^t\right).$$

This private key is a properly distributed private key for $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$ for $r = r' + t \in \mathbb{Z}_p^*$.

`Encrypt`$(params, \mathsf{ID}, M)$**:** To encrypt a message $M \in \mathbb{G}_1$ under identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$, pick a random $s \in \mathbb{Z}_p^*$ and output

$$\mathsf{CT} = \left(e(g_1, g_2)^s \cdot M, \quad g^s, \quad \left(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3\right)^s\right) \in \mathbb{G}_1 \times \mathbb{G}^2.$$

`Decrypt`$(d_{\mathsf{ID}}, \mathsf{CT})$**:** Consider an identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$. To decrypt a given ciphertext $\mathsf{CT} = (A, B, C)$ using the private key $d_{\mathsf{ID}} = (a_0, a_1, b_{k+1} \ldots, b_\ell)$, output

$$A \cdot e(a_1, C) \Big/ e(B, a_0) = M.$$

Indeed, for a valid ciphertext, we have

$$\frac{e(a_1, C)}{e(B, a_0)} = \frac{e\left(g^r, \ (h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3)^s\right)}{e\left(g^s, \ g_2^\alpha (h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3)^r\right)} = \frac{1}{e(g, g_2)^{s\alpha}} = \frac{1}{e(g_1, g_2)^s}.$$

Observe that for identities at any depth, the ciphertext contains only 3 elements and decryption takes only 2 pairings. In previous HIBE systems, ciphertext size and decryption time grow linearly in the identity depth. Also, note that $e(g_1, g_2)$ used for encryption can be precomputed (or substituted for g_2 in the system parameters) so that encryption does not require any pairings.

4.1.1 Security

We first define the security model for HIBE before proving our scheme secure under that model.

Fully Secure HIBE Systems. Like an Identity Based Encryption (IBE) system, a Hierarchical Identity Based Encryption (HIBE) system consists of four algorithms [43, 34, 5]: `Setup`, `KeyGen`, `Encrypt`, `Decrypt`. In HIBE, however, identities are vectors; a vector of dimension k represents an identity at depth k. The `Setup` algorithm generates system parameters, denoted by *params*, and a master key *master-key*. We refer to the *master-key* as the private key at depth 0 and note that an IBE system is a HIBE where all identities are at depth 1. Algorithm `KeyGen` takes as input an identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$ at depth k and the private key $d_{\mathsf{ID}|k-1}$ of the parent identity $\mathsf{ID}_{|k-1} = (\mathrm{I}_1, \ldots, \mathrm{I}_{k-1})$ at depth $k-1$, and then outputs the private key d_{ID} for identity ID. The encryption algorithm encrypts messages for an identity using *params* and the decryption algorithm decrypts ciphertexts using the private key.

Chosen ciphertext security for HIBE systems is defined under a chosen identity attack where the adversary is allowed to adaptively chose the public key on which it will be challenged. More precisely, HIBE security (IND-ID-CCA) is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

IND-ID-CCA Security Game:

Setup: The challenger $\mathcal{C}$ runs the `Setup` algorithm and gives $\mathcal{A}$ the resulting system parameters *params*, keeping the *master-key* to itself.

Phase 1: $\mathcal{A}$ adaptively issues queries $q_1, \ldots, q_m$ where query q_i is one of the following:

- Private key query $\langle \mathsf{ID}_i \rangle$. $\mathcal{C}$ responds by running algorithm `KeyGen` to generate the private key d_i corresponding to the public key $\langle \mathsf{ID}_i \rangle$ and sends d_i to $\mathcal{A}$.
- Decryption query $\langle \mathsf{ID}_i, C_i \rangle$. $\mathcal{C}$ responds by running algorithm `KeyGen` to generate the private key d corresponding to ID_i. It then runs algorithm `Decrypt` to decrypt the ciphertext C_i using the private key d and sends the resulting plaintext to $\mathcal{A}$.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is over, it outputs an identity ID^* and two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ on which it wishes to be challenged. The only restriction is that $\mathcal{A}$ did not previously issue a private key query for ID^* or a prefix of ID^*. $\mathcal{C}$ picks a random bit $b \in \{0, 1\}$ and sets the challenge ciphertext to $\mathsf{CT} = \mathtt{Encrypt}(\mathit{params}, \mathsf{ID}^*, M_b)$, which is sent to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ issues additional queries $q_{m+1}, \ldots, q_n$ where q_i is one of:

- Private key query $\langle \mathsf{ID}_i \rangle$ where $\mathsf{ID}_i \neq \mathsf{ID}^*$ and ID_i is not a prefix of ID^*.
- Decryption query $\langle C_i \rangle \neq \langle C \rangle$ for ID^* or any prefix of ID^*.

In both cases, $\mathcal{C}$ responds as in Phase 1. These queries may be adaptive.

Guess: Finally, the adversary outputs a guess $b' \in \{0, 1\}$ and wins if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-ID-CCA adversary. We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme $\mathcal{E}$ as

$$\mathrm{Adv}_{\mathcal{E},\mathcal{A}} = \left| \Pr[b = b'] - 1/2 \right|.$$

The probability is over the random bits used by the challenger and the adversary.

Canetti, Halevi, and Katz [19, 20] define a weaker notion of security in which the adversary commits ahead of time to the public key it will attack. We refer to this notion as selective identity, chosen ciphertext secure HIBE (IND-sID-CCA). The game is exactly the same as IND-ID-CCA except that the adversary $\mathcal{A}$ discloses to the challenger the target identity ID^* before the **Setup** phase. The restrictions on private key queries from phase 2 also hold in phase 1.

Definition 4.1.1. We say that a HIBE system $\mathcal{E}$ is $(t, q_{\mathsf{ID}}, q_C, \epsilon)$-secure if for any t-time IND-ID-CCA (respectively IND-sID-CCA) adversary $\mathcal{A}$ that makes at most q_{ID} chosen private key queries and at most q_C chosen decryption queries, we have that $\mathrm{Adv}_{\mathcal{E},\mathcal{A}} < \epsilon$. As shorthand, we say that $\mathcal{E}$ is $(t, q_{\mathsf{ID}}, q_C, \epsilon)$-IND-ID-CCA (resp. IND-sID-CCA) secure.

Semantic Security. As usual, we define chosen plaintext security for a HIBE system as in the preceding game, except that the adversary is not allowed to issue any decryption queries. The adversary may still issue adaptive private key queries. This security notion is termed as IND-ID-CPA (or IND-sID-CPA in the case of a selective identity adversary).

Definition 4.1.2. We say that a HIBE system $\mathcal{E}$ is $(t, q_{\mathsf{ID}}, \epsilon)$-IND-ID-CPA secure (resp. IND-sID-CPA) if $\mathcal{E}$ is $(t, q_{\mathsf{ID}}, 0, \epsilon)$-IND-ID-CCA secure (resp. IND-sID-CCA).

We now show that our HIBE scheme is selective identity secure (IND-sID-CPA) under the decisional Bilinear Diffie-Hellman Inversion assumption. We use a slightly weaker assumption called the decisional Weak BDHI as stated in Section 2.5. We later describe how to provide both chosen ciphertext security (IND-sID-CCA) and full HIBE security (IND-ID-CCA).

Theorem 4.1.1. *Let $\mathbb{G}$ be a bilinear group of prime order p. Suppose the Decision (t, ϵ, ℓ)-wBDHI assumption holds in $\mathbb{G}$. Then the previously defined ℓ-HIBE system is (t', q_S, ϵ)-selective identity, chosen plaintext* (IND-sID-CPA) *secure for arbitrary ℓ, q_S, and $t' < t - \Theta(\tau\, \ell\, q_S)$, where τ is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the ℓ-HIBE system. Using $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the Decision ℓ-wBDHI* problem in $\mathbb{G}$.

For a generator $g \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p^*$ let $y_i = g^{(\alpha^i)} \in \mathbb{G}$. Algorithm $\mathcal{B}$ is given as input a random tuple $(g, h, y_1, \ldots, y_\ell, T)$ that is either sampled from $\mathcal{P}_{wBDHI^*}$ (where $T = e(g,h)^{(\alpha^{\ell+1})}$) or from $\mathcal{R}_{wBDHI^*}$ (where T is uniform and independent in $\mathbb{G}_1^*$). Algorithm $\mathcal{B}$'s goal is to output 1 when the input tuple is sampled from $\mathcal{P}_{wBDHI^*}$ and 0 otherwise. Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a selective identity game as follows:

Initialization. The selective identity game begins with $\mathcal{A}$ first outputting an identity $\widehat{\mathsf{ID}}^* = (\mathrm{I}_1^*, \ldots, \mathrm{I}_m^*) \in (\mathbb{Z}_p^*)^m$ of depth $m \leq \ell$ that it intends to attack. If $m < \ell$ then $\mathcal{B}$ pads $\widehat{\mathsf{ID}}^*$ with $\ell - m$ zeroes on the right to obtain a vector ID^* a vector of length ℓ.

Setup. To generate the system parameters, algorithm $\mathcal{B}$ picks a random γ in $\mathbb{Z}_p^*$ and sets $g_1 = y_1 = g^\alpha$ and $g_2 = y_\ell \cdot g^\gamma = g^{\gamma + (\alpha^\ell)}$. Next, $\mathcal{B}$ picks random $\gamma_1, \ldots, \gamma_\ell$ in $\mathbb{Z}_p^*$ and sets $h_i = g^{\gamma_i} / y_{\ell - i + 1}$ for $i = 1, \ldots, \ell$. Algorithm $\mathcal{B}$ also picks a random δ in $\mathbb{Z}_p^*$ and sets $g_3 = g^\delta \cdot \prod_{i=1}^{\ell} y_{\ell-i+1}^{I_i^*}$.

Finally, $\mathcal{B}$ gives $\mathcal{A}$ the system parameters $params = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell)$. Observe that all these values are distributed uniformly and independently in $\mathbb{G}$ as required. The master key corresponding to these system parameters is $g_2^\alpha = g^{\alpha(\alpha^\ell + \gamma)} = y_{\ell+1} y_1^\gamma$, which is unknown to $\mathcal{B}$ since $\mathcal{B}$ does not have $y_{\ell+1}$.

Phase 1. $\mathcal{A}$ issues up to q_S private key queries. Consider a query for the private key corresponding to $\mathsf{ID} = (\mathrm{I}_1, \ldots, I_u) \in (\mathbb{Z}_p^*)^u$ where $u \leq \ell$. The only restriction is that ID is not $\mathsf{ID}*$ or a prefix of ID^*. This restriction ensures that there exists a $k \in \{1, \ldots, u\}$ such that $\mathrm{I}_k \neq \mathrm{I}_k^*$ (otherwise, ID would be a prefix of ID^*); we set k such that it is the smallest such index. To respond to the query, algorithm $\mathcal{B}$ first derives a private key for the identity $(\mathrm{I}_1, \ldots, \mathrm{I}_k)$ from which it then constructs a private key for the requested identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k, \ldots, \mathrm{I}_u)$.

To generate the private key for identity $(\mathrm{I}_1, \ldots, \mathrm{I}_k)$, $\mathcal{B}$ first picks a random $\tilde{r}$ in $\mathbb{Z}_p^*$. We pose $r = \frac{\alpha^k}{(\mathrm{I}_k - \mathrm{I}_k^*)} + \tilde{r} \in \mathbb{Z}_p^*$. Next, $\mathcal{B}$ generates the private key

$$\left(g_2^\alpha \cdot (h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} g_3)^r,\ \ g^r,\ \ h_{k+1}^r,\ \ \ldots,\ \ h_\ell^r \right), \tag{4.1}$$

which is a properly distributed private key for the identity $(\mathrm{I}_1, \ldots, \mathrm{I}_k)$. We show that $\mathcal{B}$ can compute all elements of this private key given the values at its disposal. We use the fact that $y_i^{(\alpha^j)} = y_{i+j}$ for any i, j.

To generate the first component of the private key, first observe that

$$(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} g_3)^r = \left(g^{\delta + \sum_{i=1}^k \mathrm{I}_i \gamma_i} \cdot \prod_{i=1}^{k-1} y_{\ell-i+1}^{(\mathrm{I}_i^* - \mathrm{I}_i)} \cdot y_{\ell-k+1}^{(\mathrm{I}_k^* - \mathrm{I}_k)} \cdot \prod_{i=k+1}^{\ell} y_{\ell-i+1}^{\mathrm{I}_i^*} \right)^r. \tag{4.2}$$

Let Z denote the product of the first, second, and fourth terms. That is,

$$Z = \left(g^{\delta + \sum_{i=1}^k \mathrm{I}_i \gamma_i} \cdot \underbrace{\prod_{i=1}^{k-1} y_{\ell-i+1}^{(\mathrm{I}_i^* - \mathrm{I}_i)}}_{=1} \cdot \prod_{i=k+1}^{\ell} y_{\ell-i+1}^{\mathrm{I}_i^*} \right)^r.$$

Note that the second term in Z equals 1 because $\mathrm{I}_i = \mathrm{I}_i^*$ for all $i < k$. One can verify that $\mathcal{B}$ can compute all the terms in Z given the values at its disposal. Next, observe that the third term in Eq (4.2), namely $y_{\ell-k+1}^{r(\mathrm{I}_k^* - \mathrm{I}_k)}$, is:

$$y_{\ell-k+1}^{r(\mathrm{I}_k^* - \mathrm{I}_k)} = y_{\ell-k+1}^{\tilde{r}(\mathrm{I}_k^* - \mathrm{I}_k)} \cdot y_{\ell-k+1}^{(\mathrm{I}_k^* - \mathrm{I}_k)\frac{\alpha^k}{(\mathrm{I}_k - \mathrm{I}_k^*)}} = y_{\ell-k+1}^{\tilde{r}(\mathrm{I}_k^* - \mathrm{I}_k)} / y_{\ell+1}.$$

Hence, the first component in the private key (4.1) is equal to:

$$g_2^{\alpha} (h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} g_3)^r = (y_{\ell+1} y_1^{\gamma}) \cdot Z \cdot (y_{\ell-k+1}^{\tilde{r}(\mathrm{I}_k^* - \mathrm{I}_k)} / y_{\ell+1}) = y_1^{\gamma} \cdot Z \cdot y_{\ell-k+1}^{\tilde{r}(\mathrm{I}_k^* - \mathrm{I}_k)}.$$

Since $y_{\ell+1}$ cancels out, all the terms in this expression are known to $\mathcal{B}$. Thus, $\mathcal{B}$ can compute the first private key component.

The second component, g^r, is $y_k^{1/(\mathrm{I}_k - \mathrm{I}_k^*)} g^{\tilde{r}}$ which $\mathcal{B}$ can compute. Similarly, the remaining elements $h_{k+1}^r, \ldots, h_\ell^r$ can be computed by $\mathcal{B}$ since they do not involve a $y_{\ell+1}$ term. Thus, $\mathcal{B}$ can derive a valid private key for $(\mathrm{I}_1, \ldots, \mathrm{I}_k)$. Algorithm $\mathcal{B}$ uses this private key to derive a private key for the descendant identity ID and gives $\mathcal{A}$ the result.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs two messages $M_0, M_1 \in \mathbb{G}_1$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ picks a random bit $b \in \{0, 1\}$ and responds with the challenge ciphertext

$$\mathsf{CT} = (M_b \cdot T \cdot e(y_1, h^{\gamma}), \quad h, \quad h^{\delta + \sum_{i=1}^{\ell} \mathrm{I}_i^* \gamma_i})$$

where h and T are from the input tuple given to $\mathcal{B}$. First note that if $h = g^c$ (for some unknown c in $\mathbb{Z}_p^*$) then

$$h^{\delta + \sum_{i=1}^{\ell} \mathrm{I}_i^* \gamma_i} = \left((\prod_{i=1}^{\ell} g^{\gamma_i \mathrm{I}_i^*}) \cdot g^{\delta} \right)^c = \left(\prod_{i=1}^{\ell} (g^{\gamma_i} / y_{\ell-i+1})^{\mathrm{I}_i^*} \cdot (g^{\delta} \prod_{i=1}^{\ell} y_{\ell-i+1}^{\mathrm{I}_i^*}) \right)^c = (h_1^{\mathrm{I}_1^*} \cdots h_\ell^{\mathrm{I}_\ell^*} g_3)^c,$$

and

$$e(g, h)^{(\alpha^{\ell+1})} \cdot e(y_1, h^{\gamma}) = \big(e(y_1, y_\ell) \cdot e(y_1, g^{\gamma}) \big)^c = e(y_1, y_\ell g^{\gamma})^c = e(g_1, g_2)^c.$$

Therefore, if $T = e(g,h)^{(\alpha^{\ell+1})}$ (i.e., when the input tuple is sampled from $\mathcal{P}_{wBDHI^*}$), then $T \cdot e(y_1, h^\gamma) = e(g_1, g_2)^c$. It follows that the challenge ciphertext is a valid encryption of M_b under the original (unpadded) identity $\mathsf{ID}^* = (\mathrm{I}_1^*, \ldots, \mathrm{I}_m^*)$ chosen by the adversary, since

$$\begin{aligned}\mathsf{CT} &= \big(\ M_b \cdot e(g_1,g_2)^c,\quad g^c,\quad (h_1^{\mathrm{I}_1^*} \cdots h_m^{\mathrm{I}_m^*} \cdots h_\ell^{\mathrm{I}_\ell^*} g_3)^c\ \big) \\ &= \big(\ M_b \cdot e(g_1,g_2)^c,\quad g^c,\quad (h_1^{\mathrm{I}_1^*} \cdots h_m^{\mathrm{I}_m^*} g_3)^c\ \big).\end{aligned}$$

On the other hand, when T is uniform and independent in $\mathbb{G}_1^*$ (when the input tuple is sampled from $\mathcal{R}_{wBDHI^*}$), CT is independent of b in the adversary's view.

Phase 2. $\mathcal{A}$ issues queries not issued in Phase 1. $\mathcal{B}$ responds as before.

Guess. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. Algorithm $\mathcal{B}$ concludes its own game by outputting a guess as follows. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning $T = e(g,h)^{(\alpha^{\ell+1})}$. Otherwise, it outputs 0 meaning T is random in $\mathbb{G}_1$.

When the input tuple is sampled from $\mathcal{P}_{wBDHI^*}$ (where $T = e(g,h)^{(\alpha^{\ell+1})}$), then $\mathcal{A}$'s view is identical to its view in a real attack game and therefore $\mathcal{A}$ satisfies $|\Pr[b = b'] - 1/2| \geq \epsilon$. When the input tuple is sampled from $\mathcal{R}_{wBDHI^*}$ (where T is uniform in $\mathbb{G}_1^*$) then $\Pr[b = b'] = 1/2$. Therefore, with g uniform in $\mathbb{G}^*$, h uniform in $\mathbb{G}$, α uniform in $\mathbb{Z}_p^*$, and T uniform in $\mathbb{G}_1$ we have that

$$\begin{aligned}\Big|\Pr\Big[\mathcal{B}\big(g,h,\vec{y}_{g,\alpha,\ell}, e(g,h)^{(\alpha^{\ell+1})}\big) = 0\Big] \quad - \quad \Pr\Big[\mathcal{B}\big(g,h,\vec{y}_{g,\alpha,\ell}, T\big) = 0\Big]\Big| \\ \geq \quad |(1/2 \pm \epsilon) - 1/2| \quad = \quad \epsilon\end{aligned}$$

as required, which completes the proof of the theorem. □

Chosen Ciphertext Security. Canetti et al. [20] show a general method of building an IND-sID-CCA secure ℓ-HIBE from a IND-sID-CPA secure $\ell+1$-HIBE. A more efficient construction is given by Boneh and Katz [13]. Applying either method to our HIBE construction results in a IND-sID-CCA secure ℓ-HIBE for arbitrary ℓ where the ciphertext length is independent of the hierarchy height.

Arbitrary Identities. We can extend our HIBE to handle arbitrary identities $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_\ell)$ with $\mathrm{I}_i \in \{0,1\}^*$ for $i = 1, \ldots, \ell$ by hashing each I_i with a collision resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$ during key generation and encryption. A standard argument shows that if the original HIBE scheme is IND-sID-CCA secure, then so is the HIBE scheme using H.

4.1.2 Full HIBE Security

Theorem 4.1.1 shows that our HIBE system is selective-ID secure without random oracles. Thus, the system is secure when the adversary commits ahead of time to the identity he intends to attack. Boneh and Boyen [5] observed that IBE systems that are selective-ID secure are also fully secure (i.e., secure against adversaries that adaptively select the identity to attack) as long as one hashes the identity prior to using it. The reduction, however, is not tight. Let $H : \{0,1\}^* \to \{0,1\}^d$ be a hash function (where, e.g., $d = 160$ bits). Assuming H is collision resistant, the reduction

introduces a 2^d multiplicative security loss factor in the standard model. When H is viewed as a random oracle, the reduction introduces a q_H multiplicative security loss factor where q_H is the number of the hash oracle queries issued by the adversary.

A similar observation applies to HIBE systems. Let $\mathcal{E}$ be a selective-ID secure HIBE of depth ℓ. Let $\mathcal{E}_H$ be an HIBE system where an identity $\mathsf{ID} = (I_1, \ldots, I_k)$ is hashed to $\mathsf{ID}_H = (H(I_1), \ldots, H(I_k))$ before using it in `KeyGen` and `Encrypt`. Then, if H is collision resistant, it follows that $\mathcal{E}_H$ is a fully secure HIBE, but the reduction introduces a loss factor of $2^{\ell d}$. In the random oracle model, $\mathcal{E}_H$ is a fully secure HIBE and the reduction introduces a loss factor of q_H^ℓ.

We remark that in the random oracle model, the public parameters are of constant size and contain only the two group elements (g, g_1); the other parameters $(g_2, g_3, h_1, \ldots, h_\ell)$ need not be specified as they can be derived by applying the oracle on a predetermined input string.

We also note that the construction of Waters [69], for a fixed depth ℓ, applied to our HIBE could give a fully secure constant ciphertext HIBE with a polynomial time reduction to the underlying complexity assumption in the standard model. The resulting private keys are much larger, namely of size $d\ell$, as opposed to ℓ in our system.

4.2 Extensions

We discuss a number of extensions to the HIBE system of the previous section.

4.2.1 Limited Delegation

Let $d_{\mathsf{ID}} = (a_0, a_1, b_k, \ldots, b_\ell)$ be the private key for the identity ID. Note that the `Decrypt` algorithm uses only the terms a_0 and a_1, and the `KeyGen` algorithm uses only the remaining terms $b_k, \ldots, b_\ell$.

By removing any number of $b_k, \ldots, b_\ell$, an identity ID at depth k can be given a restricted private key that only lets it issue private keys to descendants of bounded depth. For example, if the private key for ID only contains b_k, b_{k+1}, b_{k+2} (instead of all $b_k, \ldots, b_\ell$), then ID can only issue private keys for three generations of descendants, and those descendants' private keys will be limited even further.

We note that the security model and corresponding proof of limited delegation for our HIBE system was developed by Shacham [64].

4.2.2 HIBE with Short Private Keys

Certain applications, such as the time lock encryption (to be described in Section 4.3), are better served by using a HIBE system with short private keys rather than ciphertexts. We show how to construct a HIBE system whose private key size grows only sublinearly with hierarchy depth.

The idea is to construct a hybrid of the HIBE in Section 4.1 and the Boneh-Boyen HIBE [5]. Recall that in the former system the private key shrinks as the identity depth increases, while in the latter system the private key grows with the depth of an identity. The hybrid is based on the algebraic similarities between both systems, and exploits their opposite behavior with regard to private key size, to ensure that no private key ever contains more than $O(\sqrt{\ell})$ group elements.

Specifically, for $\omega \in [0, 1]$, the hybrid scheme achieves $O(\ell^\omega + \ell^{1-\omega})$ private key size and $O(\ell^\omega)$ ciphertext size at every level in a hierarchy of depth ℓ. The setting $\omega = 0$ corresponds to our HIBE, whereas $\omega = 1$ corresponds to the Boneh-Boyen HIBE [5]. The most efficient hybrids are obtained when $\omega \in [0, \frac{1}{2}]$. For example, when $\omega = \frac{1}{2}$, private keys and ciphertexts are of size $O(\sqrt{\ell})$.

Hybrid Scheme. As before, we assume a bilinear group $\mathbb{G}$ and a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$, where $\mathbb{G}$ and $\mathbb{G}_1$ have prime order p. Let $\ell_1 = \lceil \ell^\omega \rceil$ and $\ell_2 = \lceil \ell^{1-\omega} \rceil$. The basic idea is to partition levels of the hierarchy into ℓ_1 consecutive groups of size ℓ_2. Within each group we use the system of Section 4.1. Between groups we use the Boneh-Boyen HIBE [5].

Let $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$ be an identity of depth $k \leq \ell$. We will represent ID as a pair (k, I) where $\mathrm{I} \in (\mathbb{Z}_p^*)^{\ell_1 \times \ell_2}$ is an $\ell_1 \times \ell_2$ matrix filled using the elements $\mathrm{I}_1, \ldots, \mathrm{I}_k$ in typographic order: one row at a time starting from the top, in each row starting from the left (note that $\ell_1 \cdot \ell_2 \geq \ell \geq k$; the unfilled matrix entries are undefined). For convenience, we decompose the indices $k = 1, \ldots, \ell$ into row-column pairs (k_1, k_2) such that $k = \ell_2 \cdot (k_1 - 1) + k_2$ where $k_1, k_2 > 0$. For shorthand, we write $(k_1, k_2) = k$. It follows that in the above matrix representation of ID we have $\mathrm{I}_{(i_1,i_2)} = \mathrm{I}_i$ for all $i = 1, \ldots, k$. Or, pictorially, for an ID at the maximum depth ℓ with $\mathrm{I} = \mathrm{I}_1, \ldots, \mathrm{I}_\ell$ and $\ell = \ell_1 \ell_2$:

$$\mathrm{I} \;=\; \begin{pmatrix} \mathrm{I}_1 & \mathrm{I}_2 & \ldots & \mathrm{I}_{\ell_2} \\ \mathrm{I}_{\ell_2+1} & \mathrm{I}_{\ell_2+2} & \ldots & \mathrm{I}_{2\,\ell_2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{I}_{(\ell_1-1)\ell_2+1} & \mathrm{I}_{(\ell_1-1)\ell_2+2} & \ldots & \mathrm{I}_{\ell_1\ell_2} \end{pmatrix} \;=\; \begin{pmatrix} \mathrm{I}_{(1,1)} & \mathrm{I}_{(1,2)} & \ldots & \mathrm{I}_{(1,\ell_2)} \\ \mathrm{I}_{(2,1)} & \mathrm{I}_{(2,2)} & \ldots & \mathrm{I}_{(2,\ell_2)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{I}_{(\ell_1,1)} & \mathrm{I}_{(\ell_1,2)} & \ldots & \mathrm{I}_{(\ell_1,\ell_2)} \end{pmatrix}.$$

Using this convention, we can now describe the hybrid HIBE system as follows.

$\mathtt{Setup}(\ell, \omega)$**:** For a HIBE of maximum depth ℓ, first determine ℓ_1 and ℓ_2 as above so that $\ell \leq \ell_1 \cdot \ell_2$. Next, select a random generator g in $\mathbb{G}$, a random $\alpha \in \mathbb{Z}_p^*$, and set $g_1 = g^\alpha$. Then, pick random elements $g_2, f_1, \ldots, f_{\ell_1}, h_1, \ldots, h_{\ell_2} \in \mathbb{G}$. The public parameters *params* and the secret *master-key* are given by

$$\textit{params} = (\; g,\; g_1, g_2,\; f_1, \ldots, f_{\ell_1},\; h_1, \ldots, h_{\ell_2} \;), \qquad \textit{master-key} = g_2^\alpha.$$

$\mathtt{KeyGen}(d_{\mathsf{ID}|k-1}, \mathsf{ID})$**:** To generate private key d_{ID} for identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$ of depth $(k_1, k_2) = k \leq \ell$, where $k_1 \leq \ell_1$ and $k_2 \leq \ell_2$, pick random $r_1, \ldots, r_{k_1} \in \mathbb{Z}_p^*$, and output

$$\begin{aligned} d_{\mathsf{ID}} \;=\; \Bigg(& g_2^\alpha \cdot \Bigg(\prod_{i=1}^{k_1-1} \left(h_1^{\mathrm{I}_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}} \cdot f_i \right)^{r_i} \Bigg) \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1} \right)^{r'_{k_1}}, \\ & g^{r_1},\; \ldots,\; g^{r_{k_1-1}},\; g^{r'_{k_1}},\; h_{k_2+1}^{r'_{k_1}},\; \ldots,\; h_{\ell_2}^{r'_{k_1}} \Bigg) \;\in\; \mathbb{G}^{1+k_1+\ell_2-k_2}. \end{aligned} \tag{4.3}$$

Note that the factors $(\ldots)^{r_i}$ under the $\prod$ sign contain ℓ_2 identity terms each, whereas the last factor $(\ldots)^{r_{k_1}}$ only has k_2 such terms. The size of d_{ID} grows with k_1 and shrinks with k_2; the private key thus becomes alternatively shorter and longer as the depth of ID increases, but never exceeds $\ell_1 + \ell_2$ elements of $\mathbb{G}$.

The private key for ID can be generated with a private key for $\mathsf{ID}_{|k-1} = (\mathrm{I}_1, \ldots, \mathrm{I}_{k-1}) \in (\mathbb{Z}_p^*)^{k-1}$ as required. Decompose k as (k_1, k_2) according to our convention. There are two cases:

1. If $k-1$ is written $(k_1, k_2 - 1)$, namely k and $k-1$ have the same row index k_1, then we know

that the private key for $\mathsf{ID}_{|k-1}$ is of the form:

$$d_{\mathsf{ID}|k-1} = \Big(g_2^{\alpha} \cdot \prod_{i=1}^{k_1-1} \big(h_1^{\mathrm{I}_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}} \cdot f_i\big)^{r_i} \cdot \big(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2-1}^{\mathrm{I}_{(k_1,k_2-1)}} \cdot f_{k_1}\big)^{r_{k_1}},\ g^{r_1},$$
$$\ldots,\ g^{r_{k_1}}, h_{k_2}^{r_{k_1}},\ \ldots,\ h_{\ell_2}^{r_{k_1}} \Big) \ = \ (a_0, b_1, \ldots, b_{k_1}, c_{k_2}, \ldots, c_{\ell_2}) \ \in \mathbb{G}^{2+k_1+\ell_2-k_2}.$$

In this case, to generate d_{ID} from $d_{\mathsf{ID}|k-1}$, pick a random $r^* \in \mathbb{Z}_p^*$ and output

$$d_{\mathsf{ID}} = \Big(a_0 \cdot c_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot \big(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1}\big)^{r^*},\ \ b_1,\ \ \ldots,\ \ b_{k_1-1},\ \ b_{k_1} \cdot g^{r^*},$$
$$c_{k_2+1} \cdot h_{k_2+1}^{r^*},\ \ \ldots,\ \ c_{\ell_2} \cdot h_{\ell_2}^{r^*} \Big) \ \in \mathbb{G}^{1+k_1+\ell_2-k_2}.$$

This tuple is of the same form as Eq (4.3) where $r'_{k_1} = r_{k_1} + r^*$.

2. If the row indices differ, then necessarily $k-1 = (k_1 - 1, \ell_2)$ and $k = (k_1, 1)$, and the private key for $\mathsf{ID}_{|k-1}$ must be of the form:

$$\begin{aligned} d_{\mathsf{ID}|k-1} &= \Big(g_2^{\alpha} \cdot \prod_{i=1}^{k_1-1} \big(h_1^{\mathrm{I}_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}} \cdot f_i\big)^{r_i},\ \ g^{r_1},\ \ \ldots,\ \ g^{r_{k_1-1}} \Big) \\ &= (a_0, b_1, \ldots, b_{k_1-1}) \ \in \mathbb{G}^{k_1}. \end{aligned}$$

In this case, to generate d_{ID} from $d_{\mathsf{ID}|k-1}$, pick a random $r \in \mathbb{Z}_p^*$ and output

$$d_{\mathsf{ID}} = \Big(a_0 \cdot \big(h_1^{\mathrm{I}_{(k_1,1)}} \cdot f_{k_1}\big)^{r},\ \ b_1,\ \ \ldots,\ \ b_{k_1-1},\ \ g^{r},\ \ h_2^{r},\ \ , \ldots, h_{\ell_2}^{r} \Big) \ \in \mathbb{G}^{k_1+\ell_2}.$$

Again, this tuple conforms to Eq (4.3) in which r_{k_1} has been set to r.

`Encrypt`$(params, \mathsf{ID}, M)$**:** To encrypt a message $M \in \mathbb{G}_1$ under identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in \mathbb{Z}_p[k]$ where $k = (k_1, k_2)$, pick a random $s \in \mathbb{Z}_p^*$ and output

$$\begin{aligned} \mathsf{CT} = \Big(& e(g_1, g_2)^s \cdot M,\ \ g^s,\ \ \big(h_1^{\mathrm{I}_{(1,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(1,\ell_2)}} \cdot f_1\big)^s,\ \ \ldots\ \ , \\ & \big(h_1^{\mathrm{I}_{(k_1-1,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(k_1-1,\ell_2)}} \cdot f_{k_1-1}\big)^s,\ \ \big(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1}\big)^s \Big) \ \in \mathbb{G}_1 \times \mathbb{G}^{1+k_1}. \end{aligned}$$

`Decrypt`$(d_{\mathsf{ID}}, \mathsf{CT})$**:** Consider an identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$ with $k = (k_1, k_2)$. To decrypt a ciphertext $\mathsf{CT} = (A, B, C_1, \ldots, C_{k_1-1}, C_{k_1})$ using the private key $d_{\mathsf{ID}} = (a_0, b_1, \ldots, b_{k_1},\ c_{k_2+1}, \ldots, c_{\ell_2})$, output

$$A \cdot \prod_{i=1}^{k_1} e(b_i, C_i) \Big/ e(B, a_0) = M.$$

Note that the private key components $c_{k_2+1}, \ldots, c_{\ell_2}$ are not used for decryption.

Complexity. It is easy to see that in a hierarchy of depth ℓ, private keys at all levels contain at most $\ell_1 + \ell_2$ elements of $\mathbb{G}$, while ciphertexts contain at most $1 + \ell_1$ elements of $\mathbb{G}$ and one element of $\mathbb{G}_1$. Encryption, decryption, and one-level-down key generation, all require $O(\ell_1 + \ell_2)$ operations, or $O(\sqrt{\ell})$ for the choice $\omega = \frac{1}{2}$ as claimed. We note that the combination of having a selectable parameter ω together with the option of using an asymmetric bilinear group geared toward reducing the ciphertext or the private key size (described in Section 4.2.3), gives great flexibility toward achieving the optimal trade-off for a given application.

Security. We prove security based on the (ℓ_2)-wBDHI assumption. We note that for $\omega = 1/2$, security for a ℓ-level hierarchy is based on the $O(\sqrt{\ell})$-wBDHI assumption.

Theorem 4.2.1. *Let $\mathbb{G}$ be a bilinear group of prime order p. Consider a hybrid ℓ-HIBE system with identity hierarchy partitioned into ℓ_1 groups each of size ℓ_2. Suppose the Decision $(t, \epsilon, \ell_2 + 1)$-wBDHI assumption holds in $\mathbb{G}$. Then the hybrid ℓ-HIBE system is (t', q_S, ϵ)-selective identity, chosen plaintext* (IND-sID-CPA) *secure for arbitrary ℓ, q_S, and $t' < t - \Theta(\tau\, \ell\, q_S)$, where τ is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the hybrid ℓ-HIBE system. Using $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the Decision (ℓ_2)-wBDHI* problem in $\mathbb{G}$.

For a generator $g \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p^*$, let $y_i = g^{(\alpha^i)} \in \mathbb{G}$. Algorithm $\mathcal{B}$ is given as input a random tuple $(g, h, y_1, \ldots, y_{\ell_2}, T)$ that is either sampled from $\mathcal{P}_{wBDHI^*}$ (where $T = e(g,h)^{(\alpha^{\ell_2+1})}$) or from $\mathcal{R}_{wBDHI^*}$ (where T is uniform and independent in $\mathbb{G}_1^*$). Algorithm $\mathcal{B}$'s goal is to output 1 when the input tuple is sampled from $\mathcal{P}_{wBDHI^*}$ and 0 otherwise. Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a selective identity game as follows:

Initialization. The selective identity game begins with $\mathcal{A}$ first outputting an identity $\widehat{\mathsf{ID}}^* = (\mathrm{I}_1^*, \ldots, \mathrm{I}_m^*) \in \mathbb{Z}_p^*[m]$ of depth $m \leq \ell$ that it intends to attack. If $m < \ell$ then $\mathcal{B}$ pads $\widehat{\mathsf{ID}}^*$ with $\ell - m$ zeroes on the right to obtain a vector ID^* of length ℓ. Following our convention, we write ID^* as a pair (ℓ, I^*) where the matrix $\mathrm{I}^* \in \mathbb{Z}_p^*[\ell_1 \times \ell_2]$ is filled using the elements $\mathrm{I}_1^*, \ldots, \mathrm{I}_\ell^*$.

Setup. To generate the system parameters, algorithm $\mathcal{B}$ picks a random γ in $\mathbb{Z}_p^*$ and sets $g_1 = y_1 = g^\alpha$ and $g_2 = y_{\ell_2} \cdot g^\gamma = g^{\gamma + (\alpha^{\ell_2})}$.

Next, $\mathcal{B}$ picks random $\gamma_1, \ldots, \gamma_{\ell_2}$ in $\mathbb{Z}_p^*$ and sets $h_i = g^{\gamma_i} / y_{\ell_2 - i + 1}$ for $i = 1, \ldots, \ell_2$.

Algorithm $\mathcal{B}$ also picks random $\delta_1, \ldots, \delta_{\ell_1}$ in $\mathbb{Z}_p^*$ and sets $f_i = g^{\delta_i} \cdot \prod_{j=1}^{\ell_2} (y_{\ell_2 - j + 1})^{I^*_{(i,j)}}$ for $i = 1, \ldots, \ell_1$.

Finally, $\mathcal{B}$ gives $\mathcal{A}$ the system parameters $\mathit{params} = (g, g_1, g_2, f_1, \ldots, f_{\ell_1}, h_1, \ldots, h_{\ell_2})$. Observe that all these values are distributed uniformly and independently in $\mathbb{G}$ as required.

The master key g_4 corresponding to these system parameters is $g_4 = g^{\alpha(\alpha^{\ell_2} + \gamma)} = y_{\ell_2+1} y_1^\gamma$, which is unknown to $\mathcal{B}$ since $\mathcal{B}$ does not have y_{ℓ_2+1}.

Phase 1. $\mathcal{A}$ issues up to q_S private key queries. Consider a query for the private key corresponding to $\mathsf{ID} = (\mathrm{I}_1, \ldots, I_u) \in \mathbb{Z}_p^*[u]$ where $u \leq \ell$. The only restriction is that ID is not a prefix of ID^*. This restriction ensures that there exists a $k \in \{1, \ldots, u\}$ such that $\mathrm{I}_k \neq \mathrm{I}_k^*$ (otherwise, ID would be a prefix of ID^*); we set k such that it is the smallest such index. To respond to the

query, algorithm $\mathcal{B}$ first derives a private key for the identity $\mathsf{ID}_k = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$ from which it then constructs a private key for the requested identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k, \ldots, \mathrm{I}_u)$.

As per our convention, we write ID_k as a pair (k, I) where the matrix $\mathrm{I} \in \mathbb{Z}_p^*[\ell_1 \times \ell_2]$ is filled using the elements $\mathrm{I}_1, \ldots, \mathrm{I}_k$. Recall that our convention allows for decomposing the depth index k into a row-column pair $(k_1, k_2) = k$.

To generate the private key for the identity ID_k at depth $k = (k_1, k_2)$ where $k_1 \leq \ell_1$ and $k_2 \leq \ell_2$, algorithm $\mathcal{B}$ first picks random $r_1, \ldots, r_{k_1 - 1}, \tilde{r}_{k_1}$ in $\mathbb{Z}_p^*$. We pose $r_{k_1} = \frac{\alpha^{k_2}}{(I_{(k_1,k_2)} - I^*_{(k_1,k_2)})} + \tilde{r}_{k_1} \in \mathbb{Z}_p^*$. Next, $\mathcal{B}$ generates the private key

$$\begin{aligned} d_{\mathsf{ID}_k} \;=\; & \left(g_4 \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1}\right)^{r_{k_1}} \cdot \left(\prod_{i=1}^{k_1 - 1} \left(h_1^{\mathrm{I}_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}} \cdot f_i\right)^{r_i} \right), \right. \\ & \left. g^{r_1}, \;\; \ldots, \;\; g^{r_{k_1 - 1}}, \;\; g^{r_{k_1}}, \;\; h_{k_2+1}^{r_{k_1}}, \;\; \ldots, \;\; h_{\ell_2}^{r_{k_1}} \right) \qquad \in \; \mathbb{G}^{1 + k_1 + \ell_2 - k_2}. \end{aligned} \tag{4.4}$$

which is a properly distributed private key for the identity ID_k. We show that $\mathcal{B}$ can compute all elements of this private key given the values at its disposal. We use the fact that $y_i^{(\alpha^j)} = y_{i+j}$ for any i, j.

We first show how to generate the first component of the private key. Observe that

$$\begin{aligned} & \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1}\right)^{r_{k_1}} = \\ & \left(g^{r_{k_1} \cdot (\delta_{k_1} + \sum_{i=1}^{k_2} \mathrm{I}_{(k_1,i)} \gamma_i)} \cdot \prod_{i=1}^{k_2 - 1} y_{\ell_2 - i + 1}^{r_{k_1} \cdot (\mathrm{I}^*_{(k_1,i)} - \mathrm{I}_{(k_1,i)})} \cdot y_{\ell_2 - k_2 + 1}^{r_{k_1} \cdot (\mathrm{I}^*_{(k_1,k_2)} - \mathrm{I}_{(k_1,k_2)})} \cdot \prod_{i=k_2+1}^{\ell_2} y_{\ell_2 - i + 1}^{r_{k_1} \cdot \mathrm{I}^*_{(k_1,i)}} \right) \end{aligned} \tag{4.5}$$

Let Z denote the product of the first, second, and fourth terms. That is,

$$Z = \left(g^{r_{k_1} \cdot (\delta_{k_1} + \sum_{i=1}^{k_2} \mathrm{I}_{(k_1,i)} \gamma_i)} \right) \cdot \underbrace{\prod_{i=1}^{k_2 - 1} y_{\ell_2 - i + 1}^{r_{k_1} \cdot (\mathrm{I}^*_{(k_1,i)} - \mathrm{I}_{(k_1,i)})}}_{=1} \cdot \prod_{i=k_2+1}^{\ell_2} y_{\ell_2 - i + 1}^{r_{k_1} \cdot \mathrm{I}^*_{(k_1,i)}}.$$

Note that the second term in Z equals 1 because $\mathrm{I}_i = \mathrm{I}^*_i$ for all $i < k$ (namely, $\mathrm{I}_{(i,j)} = \mathrm{I}^*_{(i,j)}$ where both $i \leq k_1$ and $j < k_2$ hold). One can verify that $\mathcal{B}$ can compute all the terms in Z given the values at its disposal. On the other hand, $\mathcal{B}$ cannot compute the third term in Eq (4.5) by itself since it requires knowledge of $y_{\ell_2+1} = y_{\ell_2 - k_2 + 1}^{\alpha^{k_2}}$. We, however, observe that

$$y_{\ell_2 - k_2 + 1}^{r_{k_1} \cdot (\mathrm{I}^*_{(k_1,k_2)} - \mathrm{I}_{(k_1,k_2)})} = y_{\ell_2 - k_2 + 1}^{\tilde{r}_{k_1} \cdot (\mathrm{I}^*_{(k_1,k_2)} - \mathrm{I}_{(k_1,k_2)})} \Big/ y_{\ell_2 + 1} = y_{\ell_2 - k_2 + 1}^{\tilde{r}_{k_1} \cdot (\mathrm{I}^*_{(k_1,k_2)} - \mathrm{I}_{(k_1,k_2)})} \Big/ (g_4 y_1^{-\gamma}).$$

Hence, the product of the first two terms in the first component (4.4) in the private key is:

$$\begin{aligned} g_4 \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1}\right)^{r_{k_1}} &= g_4 \cdot Z \cdot y_{\ell_2 - k_2 + 1}^{\tilde{r}_{k_1} \cdot (\mathrm{I}^*_{(k_1,k_2)} - \mathrm{I}_{(k_1,k_2)})} \Big/ (g_4 y_1^{-\gamma}) \\ &= Z \cdot y_{\ell_2 - k_2 + 1}^{\tilde{r}_{k_1} \cdot (\mathrm{I}^*_{(k_1,k_2)} - \mathrm{I}_{(k_1,k_2)})} \cdot y_1^{\gamma}. \end{aligned}$$

Since g_4 cancels out from the expression on the right, all the terms in that expression are known to $\mathcal{B}$. Thus, $\mathcal{B}$ can compute the product of the first two terms in the first component of the private key (4.4). To conclude, note that the third term

$$\prod_{i=1}^{k_1-1}\left(h_1^{\mathrm{I}_{(i,1)}}\cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}}\cdot f_i\right)^{r_i}=\prod_{i=1}^{k_1-1}\left(g^{r_i\cdot(\delta_i+\sum_{j=1}^{\ell_2}\mathrm{I}_{(i,j)}\gamma_i)}\cdot\prod_{j=1}^{\ell_2}y_{\ell_2-j+1}^{r_i\cdot(\mathrm{I}^*_{(i,j)}-\mathrm{I}_{(i,j)})}\right)$$

has a similar form to Z and can be computed by $\mathcal{B}$ given the values at its disposal. Therefore, we have shown that $\mathcal{B}$ can compute the first component of the private key (4.4).

The components $g^{r_1},\ldots,g^{r_{k_1-1}}$ are easily computed by raising g to the powers of $r_1,\ldots,r_{k_1-1}$, all of which $\mathcal{B}$ knows. The component $g^{r_{k_1}}$ is simply $y_{k_2}^{1/(I_{(k_1,k_2)}-I^*_{(k_1,k_2)})}\cdot g^{\tilde{r}_{k_1}}$, which $\mathcal{B}$ can also compute. Finally, observe that

$$\begin{aligned}h_{k_2+i}^{r_{k_1}}&=\left(g^{\gamma_{k_2+i}}\big/\,y_{\ell_2-k_2-i+1}\right)^{\frac{\alpha^{k_2}}{(I_{(k_1,k_2)}-I^*_{(k_1,k_2)})}+\tilde{r}_{k_1}}\\&=\left(y_{k_2}^{\gamma_{k_2+i}}\big/\,y_{\ell_2-i+1}\right)^{\frac{1}{(I_{(k_1,k_2)}-I^*_{(k_1,k_2)})}+\tilde{r}_{k_1}},\end{aligned}$$

which $\mathcal{B}$ can compute for all $i=1,\ldots,\ell_2-k_2$ because there are no y_{ℓ_2+1} terms.

Thus, $\mathcal{B}$ can derive a valid private key for ID_k. Algorithm $\mathcal{B}$ derives a private key for the requested ID from this private key and gives $\mathcal{A}$ the result.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs two messages $M_0,M_1\in\mathbb{G}_1$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ picks a random bit $b\in\{0,1\}$ and responds with the challenge ciphertext

$$\mathsf{CT}=\left(M_b\cdot T\cdot e(y_1,h^{\gamma}),\ h,\ h^{\delta_1+\sum_{j=1}^{\ell_2}\mathrm{I}^*_{(1,j)}\gamma_j},\ h^{\delta_2+\sum_{j=1}^{\ell_2}\mathrm{I}^*_{(2,j)}\gamma_j},\ \ldots,\ h^{\delta_{\ell_1}+\sum_{j=1}^{\ell_2}\mathrm{I}^*_{(\ell_1,j)}\gamma_j}\right)$$

where h and T are from the input tuple given to $\mathcal{B}$. First note that if $h=g^c$ (for some unknown c in $\mathbb{Z}_p^*$) then

$$h^{\delta_i+\sum_{j=1}^{\ell_2}\mathrm{I}^*_{(i,j)}\gamma_j}=\left(h_1^{\mathrm{I}^*_{(i,1)}}\cdots h_{\ell_2}^{\mathrm{I}^*_{(i,\ell_2)}}\cdot f_i\right)^c$$

Therefore, if $T=e(g,h)^{(\alpha^{\ell_2+1})}$, (i.e., when the input tuple is sampled from $\mathcal{P}_{wBDHI^*}$) then the challenge ciphertext is:

$$\mathsf{CT}=\left(M_b\cdot e(g_1,g_2)^c,\ g^c,\ \left(h_1^{\mathrm{I}^*_{(1,1)}}\cdots h_{\ell_2}^{\mathrm{I}^*_{(1,\ell_2)}}\cdot f_1\right)^c,\ \ldots,\ \left(h_1^{\mathrm{I}^*_{(\ell_1,1)}}\cdots h_{\ell_2}^{\mathrm{I}^*_{(\ell_1,\ell_2)}}\cdot f_{\ell_1}\right)^c\right)$$

which is a valid encryption of M_b under the public key $\mathsf{ID}^*=(\mathrm{I}^*_1,\ldots,\mathrm{I}^*_\ell)$. On the other hand, when T is uniform and independent in $\mathbb{G}_1^*$ (when the input tuple is sampled from $\mathcal{R}_{wBDHI^*}$), then CT is independent of b in the adversary's view.

Phase 2. $\mathcal{A}$ continues to issue queries not issued in Phase 1. Algorithm $\mathcal{B}$ responds as before.

Guess. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. Algorithm $\mathcal{B}$ concludes its own game by outputting a guess as follows. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning $T = e(g,h)^{(\alpha^{\ell_2+1})}$. Otherwise, it outputs 0 meaning T is random in $\mathbb{G}_1$.

When the input tuple is sampled from $\mathcal{P}_{wBDHI^*}$ (where $T = e(g,h)^{(\alpha^{\ell_2+1})}$), then $\mathcal{A}$'s view is identical to its view in a real attack game and therefore $\mathcal{A}$ satisfies $|\Pr[b = b'] - 1/2| \geq \epsilon$. When the input tuple is sampled from $\mathcal{R}_{wBDHI^*}$ (where T is uniform in $\mathbb{G}_1^*$) then $\Pr[b = b'] = 1/2$. Therefore, with g uniform in $\mathbb{G}^*$, h uniform in $\mathbb{G}^*$ α uniform in $\mathbb{Z}_p^*$, and T uniform in $\mathbb{G}_1$ we have that

$$\left| \Pr\left[\mathcal{B}\big(g,h,\vec{y}_{g,\alpha,\ell_2}, e(g,h)^{(\alpha^{\ell_2+1})}\big) = 0\right] \;-\; \Pr\left[\mathcal{B}\big(g,h,\vec{y}_{g,\alpha,\ell_2}, T\big) = 0\right] \right| \geq \left|\left(\tfrac{1}{2} \pm \epsilon\right) - \tfrac{1}{2}\right| = \epsilon$$

as required, which completes the proof of the theorem. □

4.2.3 Asymmetric Bilinear Groups and MNT Curves.

It is often desirable to use bilinear maps $e : \mathbb{G} \times \mathbb{G}' \to \mathbb{G}_1$ where $\mathbb{G}$ and $\mathbb{G}'$ are distinct groups. Such maps let us take advantage of certain curves called MNT curves [52]. Typically, elements of the group $\mathbb{G}$ tend to afford a particularly compact representation compared to elements of $\mathbb{G}'$. This property is used for constructing short signatures [14, 6, 8]. For our system, we can use this property to shrink either the private keys or the ciphertexts.

We briefly describe how to rephrase the above HIBE in terms of asymmetric bilinear groups $e : \mathbb{G} \times \mathbb{G}' \to \mathbb{G}_1$. Recall that such groups are necessarily equipped with an efficiently computable homomorphism $\phi : \mathbb{G}' \to \mathbb{G}$. Thus, `Setup` picks all randomly selected group elements from $\mathbb{G}'$. Before these elements are used or disclosed, each element is either converted to a point in $\mathbb{G}$ using ϕ, or left alone in $\mathbb{G}'$, depending on whether this particular element is destined to appear in the first or second argument of the bilinear map $e(\cdot,\cdot)$. It is easy to see that there are no conflicts. There are two ways to perform this conversion: we can either send the private key or the ciphertext elements to $\mathbb{G}$; in one case we end up with smaller private keys, and in the other, smaller ciphertexts.

4.3 Applications

We now discuss applications of our compact HIBE system and its extensions.

4.3.1 Forward Secure Encryption

The main purpose of a forward secure encryption scheme is to guarantee that all messages encrypted before the secret key is compromised remain secret.

An elegant public key encryption scheme with forward security was proposed by Canetti, Halevi, and Katz (CHK) [19]. Let $T = 2^t$ be the number of distinct time periods in the forward secure system. When implemented with previous HIBE systems [34, 5], the CHK framework results in ciphertexts of size $O(t)$ and private keys of size $O(t^2)$. Using public updateable storage, Canetti et al. reduce private key size to $O(t)$ without affecting ciphertext length — the idea is to encrypt the private key for time period i under the public key of time period $i-1$ and store the resulting ciphertext, of size $O(t^2)$, in public storage; consequently, only one HIBE private key of size $O(t)$ is kept in private storage.

Using the HIBE system of Section 4.1 in the CHK framework, we obtain a forward secure encryption scheme with $O(1)$ ciphertext size and decryption time — independent of the number of time periods. Private keys using our basic system are of size $O(t^2)$. Alternatively, using the hybrid HIBE of Section 4.2.2 in which we set $\omega = \frac{1}{2}$, we obtain a forward secure encryption scheme with private key size $O(t^{3/2})$; in this case ciphertext size and decryption time become $O(\sqrt{t})$.

Following Canetti et al. [19], we can store most of the private key in updateable public storage in order to lessen the private storage requirement. Applied to our basic forward secure system, using $O(t^2)$ public storage we can reduce the private key size to $O(t)$ while keeping the ciphertext size constant. Using the hybrid HIBE system (for $\omega = \frac{1}{2}$), the private storage requirement can be similarly reduced to $O(\sqrt{t})$ at the cost of $O(t^{3/2})$ updateable public storage; ciphertext size in this case remains $O(\sqrt{t})$.

4.3.2 Forward Secure HIBE

Recently, a forward secure HIBE scheme was proposed by Yao et al. [71]. Their scheme essentially uses two HIBE hierarchies in the manner of Canetti et al. [19] to obtain forward security together with the ability to derive subordinate keys. Their system has ciphertexts of size $O(\ell \cdot t)$ where ℓ is the depth of the identity hierarchy and $T = 2^t$ is the number of time periods. Indeed, they pose as an open problem if a forward secure HIBE scheme with "linear" complexity is possible.

Instantiating both hierarchies in their construction with our HIBE system immediately gives a forward secure HIBE scheme with ciphertexts of size $O(1)$, which resolves this question.

4.3.3 Public Key NNL Broadcast Encryption

Broadcast encryption schemes, introduced by Fiat and Naor [30], are cryptosystems designed for the efficient broadcast of data to a dynamic group of users authorized to receive the data. Naor, Naor, and Lotspiech [54] considered broadcast encryption in the stateless receiver setting; they provided a general "subset cover" framework for such broadcast encryption schemes and gave two instances of the framework — the Complete Subtree (CS) method and the more efficient Subset Difference (SD) method. Further improvements have been proposed such as the Layered Subset Difference (LSD) [42] and the Stratified Subset Difference (SSD) [39]. In the symmetric key setting, only a "center" that possesses the secret keys can broadcast to the users. In a public key broadcast encryption system, anyone is allowed to broadcast to selected subsets of users.

Using the HIBE framework, Dodis and Fazio [27] showed how to translate the SD and LSD methods to the public key setting. For N users and r revoked users, their SD and LSD constructions based on previous HIBE systems give ciphertexts of size $O(r \cdot \log N)$, which is no better than the basic CS method. Substituting the HIBE system of Section 4.1 restores the full benefits of both SD and LSD, which results in ciphertexts of size $O(r)$.

4.3.4 Encrypting to the Future

Mont et al. [53] observed that an IBE system gives a mechanism for encrypting to the future using a trusted server. Let D be a certain date string. We view D as a public key in an IBE system. Every day, a trusted server publishes the private key corresponding to that day, which enables messages encrypted for that day to be decrypted. Methods for encrypting to the future without a trusted server were proposed by Rivest, Shamir, and Wagner [62].

One problem with the IBE timelock mechanism is that after n days have passed, the server has to publish a bulletin board with n private keys on it (one private key for each day). The amount of data on the bulletin board can be greatly reduced by using the CHK forward secure encryption scheme *in reverse*. Suppose the CHK framework is set up for a total of T time periods (using a tree of depth $\log_2 T$). To encrypt a message for day $n < T$, use the CHK public key for time period $T - n$. Similarly, on day n the trusted server publishes the CHK private key corresponding to time period $T - n$. This single private key enables anyone to derive the private keys for CHK time periods $T - n, T - n + 1, \ldots, T$. Anyone can thus decrypt messages intended for days in the range $1, \ldots, n$.

Implementing this encoding using our $O(1)$ ciphertext HIBE, the trusted server on any day only needs to publish a single private key comprising $O(\log^2 T)$ group elements. Using the hybrid HIBE system of Section 4.2.2, the private key posted by the server is further reduced to $O(\log^{3/2} T)$ group elements for ciphertexts of size $O(\sqrt{\log T})$. These parameters are much better than the IBE based mechanism [53], where the bulletin board contains as many as T group elements.

4.4 Summary and Open Problems

We presented a new HIBE system where the ciphertexts consist of three group elements and decryption only requires computing two bilinear maps, both of which are independent of the hierarchy depth. Encryption time is as efficient as other HIBE systems. For a hierarchy of depth ℓ, we proved security based on the (Weak) ℓ-BDHI assumption.

We discussed several applications of our system, including efficient forward secure encryption, an efficient public key version of the NNL broadcast encryption system, and an efficient mechanism for encrypting to the future. Our HIBE system allows for limited delegation and can be combined with the Boneh-Boyen HIBE to form a hybrid HIBE that has sublinear private key size.

We note that our selective-ID proof of security is tight. On the other hand, the proof of full security (either in the random oracle or standard model) discussed in Section 4.1.2 degrades exponentially in the hierarchy depth. The same is true for all existing HIBE systems. It is an open problem to construct a HIBE system where security does not degrade exponentially in the hierarchy depth.

Bibliography

[1] cryptology. Encyclopædia Britannica Online, June 2007. `http://www.britannica.com/eb/article-25626`.

[2] M. Bellare, J. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Proceedings of Eurocrypt '98*, volume 1403, 1998.

[3] J. Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, 1987.

[4] M. Blum. Coin flipping by telephone. In *Proceedings of Crypto '81*, 1981.

[5] D. Boneh and X. Boyen. Efficient selective-ID identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 223–38. Springer, 2004.

[6] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[7] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.

[8] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[9] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Proceedings of Crypto 2001*, volume 2139 of *LNCS*, pages 213–29. Springer, 2001.

[10] D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003. Extended abstract in *Proceedings of Crypto 2001*.

[11] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. Cryptology ePrint Archive, Report 2005/018, 2005.

[12] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In J. Killian, editor, *Proceedings of Theory of Cryptography Conference 2005*, volume 3378 of *LNCS*, pages 325–342. Springer, 2005.

[13] D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity based encryption. In *Proceedings of RSA-CT 2005*, 2005.

[14] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *Proceedings of Asiacrypt 2001*, volume 2248 of *LNCS*, pages 514–32. Springer, 2001.

[15] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. In *Topics in Algebraic and Noncommutative Geometry*, number 324 in Contemporary Mathematics. American Mathematical Society, 2003.

[16] X. Boyen and B. Waters. Compact group signatures without random oracles. In S. Vaudenay, editor, *Proceedings of Eurocrypt 2006*, volume 4004 of *LNCS*, pages 427–444. Springer, 2006.

[17] C. Cachin, J. Camenisch, J. Kilian, and J. Müller. One-round secure computation and secure autonomous mobile agents. In *27th International Colloquium on Automata, Languages and Programming (ICALP '2000)*, volume 1853 of *Lecture Notes in Computer Science*, pages 512–523. Springer-Verlag, Berlin Germany, July 2000.

[18] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *Eurocrypt '99*, pages 107–122, 1999.

[19] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In E. Biham, editor, *Proceedings of Eurocrypt 2003*, volume 2656 of *LNCS*. Springer, 2003.

[20] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 207–22. Springer, 2004.

[21] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science*, pages 41–50, Milwaukee, Wisconsin, 23–25 Oct. 1995. IEEE.

[22] J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proceedings of 26th IEEE Symposium on Foundations of Computer Science*, pages 372–382, 1985.

[23] R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In U. Maurer, editor, *Proceedings of Eurocrypt 1996*, volume 1070 of *LNCS*, pages 72–83. Springer, 1996.

[24] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications*, 8(5):481–490, Sep 1997.

[25] J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer, 2002.

[26] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In K. Kim, editor, *Proceedings of Public Key Cryptography 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, 2001.

[27] Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receivers. In J. Feigenbaum, editor, *Proceedings of the Digital Rights Management Workshop 2002*, volume 2696 of *LNCS*, pages 61–80. Springer, 2002.

[28] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *Proceedings of the Workshop on Theory and Practice in Public Key Cryptography 2005*, 2005.

[29] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, Jul 1985.

[30] A. Fiat and M. Naor. Broadcast encryption. In D. Stinson, editor, *Proceedings of Crypto 1993*, volume 773 of *LNCS*, pages 480–91. Springer, 1993.

[31] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. Odlyzko, editor, *Proceedings of Crypto 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[32] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 1–19. Springer-Verlag, May 2004.

[33] R. Gennaro, D. Micciancio, and T. Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In *ACM Computer and Communications Security (CCS) '98*, 1998.

[34] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In Y. Zheng, editor, *Proceedings of Asiacrypt 2002*, volume 2501 of *LNCS*, pages 548–66, 2002.

[35] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *Journal of Computer and System Sciences*, 60(3):592–629, 2000.

[36] O. Goldreich. *The Foundations of Cryptography - Volume 2*. Cambridge University Press, 2004.

[37] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 174–187, Toronto, Ontario, Canada, 27–29 Oct. 1986. IEEE.

[38] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 365–377. ACM Press, 1982.

[39] M. Goodrich, J. Sun, and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. In M. Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 511–27. Springer, 2004.

[40] J. Groth, R. Ostrovsky, and A. Sahai. Non-interactive zaps and new techniques for NIZK. In C. Dwork, editor, *Proceedings of Crypto 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, 2006.

[41] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In S. Vaudenay, editor, *Proceedings of Eurocrypt 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, 2006.

[42] D. Halevy and A. Shamir. The LSD broadcast encryption scheme. In M. Yung, editor, *Proceedings of Crypto 2002*, volume 2442 of *LNCS*, pages 47–60, 2002.

[43] J. Horwitz and B. Lynn. Towards hierarchical identity-based encryption. In L. Knudsen, editor, *Proceedings of Eurocrypt 2002*, volume 2332 of *LNCS*, pages 466–81. Springer, 2002.

[44] M. Jakobsson and A. Juels. Millimix: Mixing in small batches. Technical Report 99-33, Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), Oct 1999.

[45] A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of 4th Algorithmic Number Theory Symposium*, number 1838 in LNCS, pages 385–394. Springer, Jul 2000.

[46] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval (extended abstract). In *38th Annual Symposium on Foundations of Computer Science*, pages 364–373, Miami Beach, Florida, 20–22 Oct. 1997. IEEE.

[47] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In M. Naor, editor, *Proceedings of Eurocrypt 2007*, volume 4515 of *LNCS*, pages 52–78. Springer, 2007.

[48] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[49] V. Miller. Short programs for functions on curves. Unpublished manuscript, 1986.

[50] V. Miller. The weil pairing, and its efficient calculation. *J. of Cryptology*, 17(4), 2004.

[51] S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE Transactions Fundamentals*, E85-A(2):481–84, 2002.

[52] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–43, 2001.

[53] M. C. Mont, K. Harrison, and M. Sadler. The HP time vault service: exploiting IBE for timed release of confidential information. In *Proceedings of the International World Wide Web Conference 2003*, pages 160–69. ACM, 2003.

[54] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In J. Kilian, editor, *Proceedings of Crypto 2001*, volume 2139 of *LNCS*, pages 41–62. Springer, 2001.

[55] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In K. Nyberg, editor, *Proceedings of Eurocrypt 1998*, volume 1403 of *LNCS*, pages 308–318. Springer-Verlag, May 1998.

[56] P. Pallier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Proceedings of Eurocrypt 1999*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, May 1999.

[57] T. P. Pedersen. A threshold cryptosystem without a trusted party. In D. Davies, editor, *Proceedings of Eurocrypt 1991*, volume 547 of *LNCS*, pages 522–526. Springer, 1991.

[58] D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. Maurer, editor, *Proceedings of Eurocrypt 1996*, volume 1070 of *LNCS*, pages 387–398. Springer, 1996.

[59] M. Rabin. Transaction protection by beacons. *Journal of Computer and System Science*, 27(2):256–267, 1983.

[60] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 1978.

[61] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb 1978.

[62] R. Rivest, A. Shamir, and D. Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, MIT Laboratory for Computer Science, 1996.

[63] T. Sander, A. Young, and M. Yung. Non-interactive CryptoComputing for NC^1. In *Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS)*, pages 554–567, New York, NY, USA, Oct. 1999. IEEE Computer Society Press.

[64] H. Shacham. The BBG HIBE has limited delegation. Cryptology ePrint Archive, Report 2007/201, 2007.

[65] A. Shamir. Identity-based cryptosystems and signature schemes. In G. Blakley and D. Chaum, editors, *Proceedings of Crypto 1984*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.

[66] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Proceedings of Eurocrypt 1997*, volume 1233 of *LNCS*, pages 256–66. Springer, 1997.

[67] V. Shoup. Practical threshold signatures. In B. Preneel, editor, *Proceedings of Eurocrypt 2000*, volume 1807 of *LNCS*, pages 207–220. Springer, 2000.

[68] J. van de Graaf and R. Peralta. A simple and secure way to show validity of your private key. In *Proceedings of Crypto '87*, 1987.

[69] B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *Proceedings of Eurocrypt 2005*, LNCS. Springer, 2005.

[70] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Symposium on Foundations of Computer Science (FOCS)*, pages 160–164. IEEE Computer Society Press, 1982.

[71] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In B. Pfitzmann, editor, *Proceedings of the ACM Conference on Computer and Communications Security 2004*, pages 354–63, 2004.