Attribute-Based Encryption for Boolean Formulas

Lucas Kowalczyk

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2019
ABSTRACT

Attribute-Based Encryption for Boolean Formulas

Lucas Kowalczyk

We present attribute-based encryption (ABE) schemes for Boolean formulas that are adaptively secure under simple assumptions. Notably, our KP-ABE scheme enjoys a ciphertext size that is linear in the attribute vector length and independent of the formula size (even when attributes can be used multiple times), and we achieve an analogous result for CP-ABE. This resolves the central open problem in attribute-based encryption posed by Lewko and Waters. Along the way, we develop a theory of modular design for unbounded ABE schemes and answer an open question regarding the adaptive security of Yao’s Secret Sharing scheme for $\text{NC}^1$ circuits.
Contents

Acknowledgements iii

1 Introduction 1

1.1 Properties of ABE Schemes 2

1.2 Existing Constructions 9

1.3 Our Contribution 10

1.4 Organization 11

2 Preliminaries 12

2.1 Monotone Boolean formulas and $\text{NC}^1$ 12

2.2 Secret Sharing 14

2.3 Attribute-Based Encryption 16

2.4 Prime-Order Asymmetric Bilinear Groups and the Matrix Diffie-Hellman Assumption 17

2.5 Composite-Order Asymmetric Bilinear Groups and Subgroup Decision Assumptions 19

3 Unbounded ABE 21

3.1 Textbook ABE Construction 21
3.2 Meta-Attribute Encoding Technique ........................................... 24
3.3 Our Contribution ................................................................. 25
3.4 Entropy Expansion Lemma ..................................................... 30
3.5 Entropy Expansion Lemma, Revisited ....................................... 35
3.6 Concluding Remarks ............................................................ 39

4 Adaptive ABE ................................................................. 41
  4.1 Overview ................................................................. 41
  4.2 Piecewise Guessing Framework for Adaptive Security ............. 47
  4.3 Pebbling Strategy for NC$^1$ .............................................. 53
  4.4 Core Adaptive Security Component ...................................... 65
  4.5 KP-ABE for Boolean Formulas .......................................... 78
  4.6 CP-ABE for Boolean Formulas .......................................... 88

5 Putting It All Together ...................................................... 102
  5.1 Unbounded KP-ABE for Boolean formulas ............................... 102

Conclusion ............................................................................ 116
  5.2 Summary of Results ......................................................... 116
  5.3 Future Directions ............................................................. 117

Bibliography ........................................................................ 119
Acknowledgements

We spent all this time searching for ABE results, but maybe the real results were the friends we made along the way?

I would like to thank my parents for being supportive of my desire to spend so much time thinking about silly math problems. I would like to thank Allison Bishop and Tal Malkin for being great advisors. Allison has been a wonderful research mentor, a close friend, and the single greatest positive influence on my experience in graduate school; Tal has taught me the most about how to be a good teacher, and I will be eternally indebted to her for being my first introduction to theoretical computer science and cryptography. I would like to express my gratitude to all my coauthors over the last few years, on works both in and outside of this thesis: Allison Bishop, Jie Chen, Georg Fuchsbauer, Romain Gay, Junquin Gong, Abhishek Jain, Jiahui Liu, Tal Malkin, Kailash Meiyappan, Claudio Orlandi, Valerio Pastro, Mariana Raykova, Kevin Shi, Jonathan Ullman, Hoeteck Wee, Daniel Wichs, and Mark Zhandry. This work and most of my time in graduate school was supported in part by an NSF Graduate Research Fellowship (DGE-16-44869). Lastly, I’d like to thank Jahson and the good men at Milano Market \(^1\) for more than eight years of delicious sandwiches and kind hospitality.

\(^1\)Milano Market, 2892 Broadway, New York, NY 10025
To my grandfather
Chapter 1

Introduction

Attribute-Based Encryption (ABE) is a type of public key encryption which allows for fine-grained access control to encrypted data. In Ciphertext-Policy ABE (CP-ABE), secret keys are associated with attributes like

\[(\text{Department} : \text{ComputerScience}), (\text{Clearance} : \text{Secret})\],

and ciphertexts are associated with access policies that take in a set of attributes from a key and return True if the key is capable of decrypting ciphertexts associated with that policy and return False otherwise. An example of a Boolean formula access policy that evaluates to False over the previously mentioned example attributes is:

\[((\text{Department} : \text{Mathematics}) \ \text{AND} \ (\text{Clearance} : \text{Secret}))\]

OR

\[((\text{Department} : \text{ComputerScience}) \ \text{AND} \ (\text{Clearance} : \text{TopSecret}))\]

Security guarantees that (potentially colluding) users without authorized keys are not able to learn anything about an encrypted message. For example, a user with a key for the previously mentioned attributes (which do not satisfy the previously mentioned policy) should not be able to decrypt a ciphertext for the example policy
even if they collude with another unauthorized user holding a key for the following attributes:

(Department : English), (Clearance : TopSecret)

even though the union of the two user’s attributes does satisfy the access policy. Preventing such a collusion from gaining access makes constructing ABE a highly nontrivial problem. ABE therefore presents a vision for non-interactive and cryptographically-enforced information access control that seems increasingly desirabe in today’s highly-networked world. A dual notion of ABE called Key-Policy ABE (KP-ABE) swaps the roles of access policies and attributes to be associated with the secret keys and ciphertexts respectively. In this work, we present constructions of CP-ABE and KP-ABE that are the first to simultaneously achieve several ideal properties.

1.1 Properties of ABE Schemes

We now introduce these properties informally, and will later formally define them in Chapter 2:

Security Model

Similar to public-key encryption, we define security for attribute-based encryption in the form of a game between a challenger and an adversary, where the adversary tries to learn some information about the contents of a challenge ciphertext for which it is unauthorized (unauthorized meaning “not supposed to be accessible”). In the game
defining security for standard public-key encryption, this lack of access is modeled by denying the adversary the secret-key which would enable straightforward decryption of the challenge ciphertext. In this sense, the game definition classifies all unauthorized adversaries the same: they are defined by not having access to the secret key. In attribute-based encryption, we again would like to define an unauthorized adversary as one without a secret-key that decrypts the challenge ciphertext, but notice that in the ABE setting there can be many secret keys which don’t decrypt the challenge ciphertext (but may decrypt other ciphertexts, depending on the attribute / policy combination of the key / ciphertext). We would like to prove that the challenge ciphertext retains its confidentiality even if the adversary has access to such an unauthorized key. Furthermore, we would like to model the fact that multiple unauthorized adversaries should not be able to combine their (unauthorized) keys to decrypt a challenge ciphertext. Finally, we would ideally like to capture the fact that a band of adversaries could attain their keys and choose what kind of ciphertext they would like to attack in an adaptive manner. That is, the specific attributes and policies that define the challenge ciphertext and unauthorized secret keys may be chosen in response to seeing previously chosen ciphertext / secret keys.

Achieving the ideal notion of provable security for ABE constructions is therefore much more difficult than in the standard public-key case, since security proofs must deal with multiple colluding adversaries who have access to potentially complex access policies interacting with attributes that are adaptively chosen. One way to make our job easier is to consider restricted notions of security. For KP-ABE \(^1\), the notion

\(^1\)As noted before, the definitions of Key-Policy (KP) and Ciphertext-Policy (CP) ABE are
of *selective security* requires the adversary to commit to a target set of attributes for the challenge ciphertext that will be attacked at the start of the security game. The earliest constructions of ABE using bilinear groups were proven secure in this model [Goy+06; Wat11]. A slightly stronger notion of *semi-adaptive security* [CW14] still requires the adversary to commit to a target set of attributes, but allows the adversary to see the public parameters first. These notions are obviously not realistic attack scenarios, so a KP-ABE scheme would ideally satisfy the notion of *adaptive security*\(^2\), where the challenge attributes can be chosen at any time in an adaptive manner (in response to public parameters and any amount of secret keys received). The first construction of ABE achieving adaptive security appeared in [Lew+10], employing the dual system encryption methodology [Wat09] in its security reduction.

**Underlying Assumption**

In cryptography, an adversary’s performance in a security game is often analyzed in the *standard model* where its computational power is restricted, and a hypothetical successful adversary is shown to correspond to an attack on a computational hardness assumption. Security may be reduced to different computational hardness assumptions with varying degrees of difficulty (often directly related to the general degree of confidence in the assumption). One way to make provable security of ABE schemes easier to achieve is to reduce security to parameterized assumptions like

---

\(^2\)Also called “full security”
q-type assumptions, where the size of the elements included in the assumption’s challenge grows with some property of the adversary. q-type assumptions were used in the ABE constructions of [Wat11; LW12] to prove adaptive security. For example, here is the “Source Group q-Parallel BDHE Assumption in a Subgroup” assumption used in [LW12]:

**Definition 1 (Source Group q-Parallel BDHE Assumption in a Subgroup).** Given group \( \mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \), \( g_1 \leftarrow G_{p_1}, g_2 \leftarrow G_{p_2}, g_3 \leftarrow G_{p_3}, c, d, f, b_1, ..., b_q \leftarrow \mathbb{Z}_n \), the adversary will be given:

\[
D := (\mathbb{G}, g_1, g_2, g_3, g_1^f, g_2^d, g_3^c, g_2^{c_i}, g_2^{c_{q+1}}, ..., g_2^{c_{2q}}),
\]

\[
g_2^j \forall i \in [2q] \setminus \{q + 1\}, j \in [q]
\]

\[
g_2^{dfb} \forall j \in [q], g_2^{efb} \forall j, j' \in [q] \text{ such that } j \neq j'
\]

We additionally define:

\[
T_0 := g_2^{c_{q+1}}, \quad T_1 \leftarrow G_{p_2}
\]

and define:

\[
\text{Adv}_{G, A}(\lambda) := |\text{Pr}[A(D, T_0) = 1] - \text{Pr}[A(D, T_1) = 1]|\]

The Source Group q-Parallel BDHE Assumption in a Subgroup states that \( \text{Adv}_{G, A}(\lambda) \) is a negligible function of \( \lambda \).

The sheer descriptive size and complexity of assumptions like this serves as a serious barrier for serious mathematical cryptanalysis. As a result, the security of
these complicated and so-called dynamic assumptions is not well-understood, and the assumptions are often closely related to the schemes in which they are used. For example, the assumption may include a number of group elements that scales with the complexity or number of queries made by the adversary in the security proof. Furthermore, it is known that many $q$-type assumptions become stronger as $q$ grows [Che06]. So, we would ideally like to reduce security of ABE constructions to simpler assumptions of a static (fixed) size, like the Decisional Linear Assumption (DLIN), Symmetric External Diffie-Hellman Assumption (SXDH), or the more general $k$-Linear Assumption ($k$-Lin). These standard assumptions are defined later in Chapter 2.

We remark that besides the standard model, there exist more restrictive paradigms like the random oracle model (ROM) [BR93] and the generic group model (GGM) [Sho97] that put further limits on the adversary besides its computational power. These restrictive models have enabled constructions of ABE [LW11a; BSW07] at points in this security/efficiency design space that were previously unachievable. At the same time, there exist cryptographic constructions that are provably secure in restricted models like the ROM and GGM which are provably *insecure* in the standard model. As such, we view proofs in such idealized models more as heuristics and strive for constructions with provable security in the standard model.
Supported Policies

A natural class of access policies one would like to be able to support in an ABE construction is that of general Boolean formulas (equivalent to the circuit class $\mathbf{NC}^1$, as discussed later in Chapter 2). Unfortunately, it has proven tremendously difficult to construct ABE for Boolean formulas with the previously discussed ideal properties of adaptive security under static assumptions in the standard model. Nearly all constructions satisfying these properties \(^3\) suffer from a “one-use restriction”. That is, they only natively support read-once Boolean formulas\(^4\). One way to extend such constructions to support formulas that use attributes more than once is to use copies of new “meta-attributes” that stand for each use of the original attribute, and are handled as a group. The downside of this approach is that it destroys the compactness of the construction – for KP-ABE, the size of the ciphertexts no longer depends on just the attribute set of the ciphertext, but also on the complexity of the formulas that the scheme supports (namely, the ciphertexts grow with the maximum number of attribute uses in any formula supported). Ciphertexts associated with an attribute-vector\(^5\) of size $n$ in a scheme like [Lew+10] where policies can reuse attributes at most $k$ times are of size $O(n \cdot k)$. For efficiency and aesthetic reasons, we prefer constructions of attribute-based encryption which are compact. That is, either the key or ciphertext size grows only with $n$ and is independent of the complexity of the

---

\(^3\)with the exception of [Tak17], [Kow+18], which will be discussed later

\(^4\)formulas where attributes are used at most once in inputs

\(^5\)Let $\{n\}$ be the set of all supported attributes. Some works associate ciphertexts with a set $S \subseteq \{n\}$, in which case the corresponding attribute-vector $x \in \{0, 1\}^n$ corresponds to the characteristic vector of $S$. 

7
supported policy set.

As mentioned in Footnote 3, the constructions of [Tak17; Kow+18] are the only KP-ABE schemes proved adaptively secure from static assumptions featuring ciphertexts that do not grow directly with the number of attribute uses $k$, but unfortunately, their ciphertexts still have a dependence on the set of allowed policies. Specifically, their constructions support policies in the form of monotone span programs (a computational model that can directly emulate Boolean formulas), and ciphertexts are of size $O(n + r)$, where $r$ is the maximum number of columns in any policy matrix supported (this is the policy dependency). For Boolean formulas (which have fan-in 2, fan-out 1), standard techniques [LW11a] to translate the formula into a policy matrix result in $r$ being equal to the number of AND gates in the formula. [Kow+18] provide an argument that their ciphertexts of size $O(n + r)$ are asymptotically incomparable to and in some cases a distinct improvement over the $O(n \cdot k)$-size ciphertexts of all other known ABE schemes proved adaptively secure under static assumptions, but again, these two exception constructions fall short of true compactness due to the policy-dependence of $r$ that bounds the ciphertext size.

Finally, we note that ABE supporting more expressive policies than Boolean formulas is possible if one is willing to sacrifice these ideal properties of adaptive security under a simple assumption. The construction of [GVW13] uses lattice technology to support circuit access policies (where fan-out is allowed to be greater than 1) rather than Boolean formulas or monotone span programs, which makes it more expressive than any known bilinear scheme. However, it was proven only selectively secure under the standard Learning With Errors (LWE) assumption. The construction of [BV16]
later extended this to semi-adaptive security for polynomial-sized circuit access policies from LWE. Circuit access policies are also supported by the constructions of [Gar+13] and [Gar+14]. These schemes are proven selectively secure and adaptively secure, respectively, under computational hardness assumptions in multilinear groups. Lastly, we note that the fully secure general functional encryption scheme in [Wat14], which relies on indistinguishability obfuscation, can also be specialized to the ABE setting. Unfortunately, at this time, the security of candidates for indistinguishability obfuscation and multilinear group hardness assumptions is not well-understood. We note that proving adaptive security for a ABE scheme supporting polynomial-sized circuits from LWE or an assumption on bilinear maps remains an open problem.

1.2 Existing Constructions

In summary, we desire ABE schemes for the expressive class of general Boolean formulas that:

(1) achieve adaptive security (with polynomial security loss);

(2) rely on simple static hardness assumptions in the standard model; and

(3) enjoy a compactness property, where the size of either the key or ciphertext grows only with the size of the attribute vector and is independent of the complexity of the supported policy class (even for complex policies that refer to each attribute many times).

but no known scheme simultaneously satisfies (1)-(3) despite many constructions achieving different combinations of these properties [Goy+06; Lew+10; OT10; LW12].
(see Figure 1.1) and unifying frameworks improving our understanding of the design and analysis of such schemes: [Att14; Wee14; CGW15; Att16; AC17]; Indeed, constructing ABE that simultaneously satisfies properties (1)-(3) is widely regarded as one of the main open problems in attribute-based encryption [Lew12].

1.3 Our Contribution

We present the first KP-ABE and CP-ABE schemes for Boolean formulas that simultaneously realize properties (1)-(3). Our KP-ABE scheme achieves ciphertext size that is linear in the attribute vector length and independent of the policy size even in the many-use setting; the same holds for the key size in our CP-ABE. Both schemes achieve adaptive security under the $k$-Lin assumption in asymmetric prime-order bilinear groups in the standard model with polynomial security loss.

Along the way, we develop a modular understanding of “unbounded” ABE, where we additionally require that the runtime of the construction’s Setup algorithm’s depends only on the security parameter$^6$. We use this understanding to present an unbounded variant of our compact KP-ABE scheme with constant-size public pa-

---

$^6$Note that this necessarily results in public parameters which are of constant size relative to the attribute-vector length and policy complexity.
rameters (see Figure 1.2), which is therefore the first unbounded adaptively secure scheme for Boolean formulas proved from a static assumption.

As an immediate corollary of our results, we note that we obtain delegation schemes for $\mathsf{NC}^1$ circuits with public verifiability and adaptive soundness under the $k$-Lin assumption [PRV12; LW12; CW14].

### 1.4 Organization

In Chapter 2 we give background on Boolean formulas, $\mathsf{NC}^1$ circuits, secret sharing, the definitions of adaptive security for attribute-based encryption we will use, and the computational hardness assumptions we will reduce security to. In Chapter 3, we present tools for achieving unbounded attribute-based encryption from bounded constructions. In Chapter 4, we prove the adaptive security of a KP-ABE and a CP-ABE scheme for Boolean formulas from the static $k$-Lin assumption. In Chapter 5, we apply our final tool from Chapter 3 to the KP-ABE scheme from Chapter 4 to obtain a construction of unbounded KP-ABE for Boolean formulas proved adaptively secure from $k$-Lin. Finally, in our Conclusion we summarize our results and discuss future directions for research.
Chapter 2

Preliminaries

Notation.

We denote by $s \leftarrow S$ the fact that $s$ is picked uniformly at random from a finite set $S$. By PPT, we denote a probabilistic polynomial-time algorithm. Throughout this paper, we use $1^k$ as the security parameter. We use lower case boldface to denote (column) vectors and upper case boldcase to denote matrices. We use $\equiv$ to denote two distributions being identically distributed, and $\approx_c$ to denote two distributions being computationally indistinguishable. For any two finite sets (also including spaces and groups) $S_1$ and $S_2$, the notation \( S_1 \approx_c S_2 \) means the uniform distributions over them are computationally indistinguishable.

2.1 Monotone Boolean formulas and NC$^1$

Monotone Boolean formulas

A monotone Boolean formula $f : \{0,1\}^n \rightarrow \{0,1\}$ is specified by a directed acyclic graph (DAG) with three kinds of nodes: input gate nodes, gate nodes, and a single output node. Input nodes have in-degree 0 and out-degree 1, AND/OR nodes have in-degree (fan-in) 2 and out-degree (fan-out) 1, and the output node has in-degree
1 and out-degree 0. We number the edges (wires) 1, 2, \ldots, m, and each gate node is defined by a tuple \((g, a_g, b_g, c_g)\) where \(g : \{0, 1\}^2 \rightarrow \{0, 1\}\) is either AND or OR, \(a_g, b_g\) are the incoming wires, \(c_g\) is the outgoing wire and \(a_g, b_g < c_g\). The size of a formula \(m\) is the number of edges in the underlying DAG and the depth of a formula \(d\) is the length of the longest path from the output node.

**NC^1 and log-depth formulas**

A standard fact from complexity theory tells us that the circuit complexity class monotone NC^1 is captured by monotone Boolean formulas of log-depth and fan-in two. This follows from the fact that we can transform any depth \(d\) circuit with fan-in two and unbounded fan-out into an equivalent circuit with fan-in two and fan-out one (for all gate nodes) of the same depth, and a \(2^d\) blow-up in the size. To see this, note that one can start with the root gate of an NC^1 circuit and work downward by each level of depth. For each gate \(g\) considered at depth \(i\), if either of its two input wires are coming from the output wire of a gate (at depth \(i - 1\)) with more than one output wire, then create a new copy of the gate at depth \(i - 1\) with a single output wire going to \(g\) (note that this copy may increase the output wire multiplicity of gates at depth strictly lower than \(i - 1\)). This procedure preserves the functionality of the original circuit, and has the result that at its end, each gate in the circuit has input wires which come from gates with output multiplicity 1. The procedure does not increase the depth of the circuit (any duplicated gates are added at a level that already exists), so the new circuit is a formula (all gates have fan-out 1) of depth \(d\)
with fan-in 2, so its size is at most $2^d$. $d$ is logarithmic in the size of the input for \( \text{NC}^1 \) circuits, so the blowup from this procedure is polynomial in \( n \). Hence we will consider the class \( \text{NC}^1 \) as a set of Boolean formulas (where gates have fan-in 2 and fan-out 1) of depth \( O(\log n) \) and refer to \( f \in \text{NC}^1 \) formulas.

\section*{2.2 Secret Sharing}

A secret sharing scheme is a pair of algorithms (\texttt{share, reconstruct}) where \texttt{share} on input \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) and \( \mu \in \mathbb{Z}_p \) outputs \( \mu_1, \ldots, \mu_m \in \mathbb{Z}_p \) together with \( \rho : [m] \rightarrow \{0, 1, \ldots, n\} \).

- Correctness stipulates that for every \( x \in \{0, 1\}^n \) such that \( f(x) = 1 \), we have

\[
\text{reconstruct}(f, x, \{\mu_j\}_{\rho(j)=0 \lor x_{\rho(j)}=1}) = \mu.
\]

- Security stipulates that for every \( x \in \{0, 1\}^n \) such that \( f(x) = 0 \), the shares

\[
\{\mu_j\}_{\rho(j)=0 \lor x_{\rho(j)}=1}
\]

perfectly hide \( \mu \) (that is, their distribution is information-theoretically independent of \( \mu \)).

Note the inclusion of \( \rho(j) = 0 \) in both correctness and security, allowing for “common share” that are always available for use in reconstruction. All the secret sharing schemes in this work will in fact be linear (in the standard sense): \texttt{share} computes a
linear function of the secret \( \mu \) and randomness over \( \mathbb{Z}_p \), and \texttt{reconstruct} computes a linear function of the shares over \( \mathbb{Z}_p \), that is, \( \mu = \sum_{\rho(j)=0 \lor x_{\rho(j)}=1} \omega_j \mu_j \).

We describe a simple information-theoretic \texttt{share} procedure for Boolean formulas (fan-in 2, fan-out 1) that can be used in the \cite{Lewe10} ABE construction lineage in Figure 4.3. The \texttt{reconstruct} procedure works in the natural way: given a set of shares corresponding to input nodes where \( x_i = 1 \) for an \( x \) such that \( f(x) = 1 \), the reconstruct procedure works upward, applying the appropriate (linear) operation at each gate to get the output wire label value, eventually reaching the root gate for which this value is the secret \( v \). \cite{Beit96} shows that finding the linear coefficients \( c_j \) such that \( v = \sum_{x_{\rho(j)}=1} c_j v_j \) can be done in polynomial time and that the shares \( \{v_j\}_{x_i=1} \) for an \( x \) such that \( f(x) = 0 \) perfectly hide \( v \). We remark that the output of share

\begin{center}
\begin{tabular}{l}
\textbf{share}(f, v):
\end{tabular}
\end{center}

\textit{Input:} A formula \( f : \{0,1\}^n \rightarrow \{0,1\} \) of size \( m \) and a secret \( v \in \mathbb{Z}_p \).

1. For root gate \( g^* \) with output wire \( c \), set \( c = v \). Then proceeding from the root gate and working downward,
   a) For each AND gate \( g \) with input wires \( a, b \) and output wire \( c \) with label \( c \), define \( a = c + z \) for an independently random \( z \leftarrow \mathbb{Z}_p \) and \( b = -z \).
   b) For each OR gate \( g \) with input wires \( a, b \) and output wire \( c \) with label \( c \), define \( a = c \) and \( b = c \).
2. For each input node with output wire \( j \) and label \( j \) associated to variable \( x_i \), define \( v_j = j \) and set \( \rho(j) = i \).
3. Output \( \{v_j\}, \rho \).

Figure 2.1: Information-theoretic linear secret sharing scheme \texttt{share} for Boolean formulas

satisfies \(|\{v_j\}| = m' \leq m\), the size of the formula, since the output consists of one \( v_j \) for each input node.
2.3 Attribute-Based Encryption

An attribute-based encryption (ABE) scheme for a predicate \( \text{pred}(\cdot, \cdot) \) consists of four algorithms (\( \text{Setup}, \text{Enc}, \text{KeyGen}, \text{Dec} \)):

\[
\text{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{M}) \rightarrow (\text{mpk}, \text{msk}). \quad \text{The setup algorithm gets as input the security parameter } \lambda, \text{the attribute universe } \mathcal{X}, \text{the predicate universe } \mathcal{Y}, \text{the message space } \mathcal{M} \text{ and outputs the public parameter } \text{mpk}, \text{and the master key } \text{msk}.
\]

\[
\text{Enc}(\text{mpk}, x, M) \rightarrow \text{ct}_x. \quad \text{The encryption algorithm gets as input mpk, an attribute } x \in \mathcal{X} \text{ and a message } M \in \mathcal{M}. \text{ It outputs a ciphertext } \text{ct}_x. \text{ Note that } x \text{ is public given } \text{ct}_x.
\]

\[
\text{KeyGen}(\text{mpk}, \text{msk}, y) \rightarrow \text{sk}_y. \quad \text{The key generation algorithm gets as input msk and a value } y \in \mathcal{Y}. \text{ It outputs a secret key } \text{sk}_y. \text{ Note that } y \text{ is public given } \text{sk}_y.
\]

\[
\text{Dec}(\text{mpk}, \text{sk}_y, \text{ct}_x) \rightarrow M. \quad \text{The decryption algorithm gets as input sk}_y \text{ and } \text{ct}_x \text{ such that } \text{pred}(x, y) = 1. \text{ It outputs a message } M.
\]

Correctness.

We require that for all \((x, y) \in \mathcal{X} \times \mathcal{Y}\) such that \(\text{pred}(x, y) = 1\) and all \(M \in \mathcal{M}\),

\[
\Pr[\text{Dec}(\text{mpk}, \text{sk}_y, \text{Enc}(\text{mpk}, x, M)) = M] = 1,
\]

where the probability is taken over \((\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{M}), \text{sk}_y \leftarrow \text{KeyGen}(\text{mpk}, \text{msk}, y), \) and the coins of \(\text{Enc}\).
Security definition.

For a stateful adversary $\mathcal{A}$, we define the advantage function

$$\text{Adv}^\text{ABE}_\mathcal{A}(\lambda) := \Pr\left[ b = b' : \begin{array}{l} (\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{M}); \\
(x^*, M_0, M_1) \leftarrow \mathcal{A}^\text{KeyGen}(\text{msk}, \cdot)(\text{mpk}); \\
b \leftarrow \{0, 1\}; \text{ct}_{x^*} \leftarrow \text{Enc}(\text{mpk}, x^*, M_0); \\
b' \leftarrow \mathcal{A}^\text{KeyGen}(\text{msk}, \cdot)(\text{ct}_{x^*}) \\
\end{array}\right] \right. - \frac{1}{2}$$

with the restriction that all queries $y$ that $\mathcal{A}$ makes to KeyGen(msk, ·) satisfy pred$(x^*, y) = 0$ (that is, sk$_y$ does not decrypt ct$_{x^*}$). An ABE scheme is adaptively secure if for all PPT adversaries $\mathcal{A}$, the advantage $\text{Adv}^\text{ABE}_\mathcal{A}(\lambda)$ is a negligible function in $\lambda$.

### 2.4 Prime-Order Asymmetric Bilinear Groups

and the Matrix Diffie-Hellman Assumption

A generator $\mathcal{G}$ takes as input a security parameter $\lambda$ and outputs a group description $\mathcal{G} := (p, G_1, G_2, G_T, e)$, where $p$ is a prime of $\Theta(\lambda)$ bits, $G_1$, $G_2$ and $G_T$ are cyclic groups of order $p$, and $e : G_1 \times G_2 \rightarrow G_T$ is a non-degenerate bilinear map. We require that the group operations in $G_1$, $G_2$ and $G_T$ as well the bilinear map $e$ are computable in deterministic polynomial time with respect to $\lambda$. Let $g \in G_1$, $h \in G_2$ and $g_T = e(g, h) \in G_T$ be the respective generators. We employ the implicit representation of group elements: for a matrix $\mathbf{M}$ over $\mathbb{Z}_p$, we define $[\mathbf{M}]_1 := g^\mathbf{M}$, $[\mathbf{M}]_2 := h^\mathbf{M}$, $[\mathbf{M}]_T := g_T^\mathbf{M}$, where exponentiation is carried out component-wise. Also, given $[\mathbf{A}]_1, [\mathbf{B}]_2$, we
let $c([A], [B]) = [AB]_T$.

We define the matrix Diffie-Hellman (MDDH) assumption on $G_1$ [Esc+13]:

**Definition 2 (MDDH$_{k,\ell}^m$ Assumption).** Let $\ell > k \geq 1$ and $m \geq 1$. We say that the MDDH$_{k,\ell}^m$ assumption holds if for all PPT adversaries $A$, the following advantage function is negligible in $\lambda$.

$$
\text{Adv}^{\text{MDDH}_{k,\ell}^m}_A(\lambda) := |\Pr[A(G, [M]_1, [MS]_1) = 1] - \Pr[A(G, [M]_1, [U]_1) = 1]|
$$

where $M \leftarrow_r \mathbb{Z}_p^{\ell \times k}$, $S \leftarrow_r \mathbb{Z}_p^{k \times m}$ and $U \leftarrow_r \mathbb{Z}_p^{\ell \times m}$.

The MDDH assumption on $G_2$ can be defined in an analogous way. Escala et al. [Esc+13] showed that

$$k\text{-Lin} \Rightarrow \text{MDDH}_{k,k+1}^1 \Rightarrow \text{MDDH}_{k,\ell}^m \forall \ell > k, m \geq 1$$

with a tight security reduction (that is, $\text{Adv}^{\text{MDDH}_{k,\ell}^m}_A(\lambda) = \text{Adv}^{k\text{-Lin}}_A(\lambda)$). Henceforth, we will use MDDH$_k$ to denote MDDH$_{k,k+1}^1$.

We note that the MDDH assumption is a generalization of the $k$-Lin Assumption, such that the $k$-Lin Assumption is equivalent to the MDDH$_{k,k+1}^1$ Assumption as defined above. Further, the $k$-Lin Assumption itself is a generalization of several standard assumptions: setting $k = 1$ yields the Symmetric External Diffie-Hellman Assumption (SXDH), while setting $k = 2$ yields the Decisional Linear Assumption (DLIN).
**Definition 3** (k-Lin Assumption). Let $k \geq 1$. We say that the k-Lin Assumption holds if for all PPT adversaries $A$, the following advantage function is negligible in $\lambda$.

$$\text{Adv}^{k\text{-lin}}_A(\lambda) := \text{Adv}^{\text{MDDH}_k}_{A,k+1}(\lambda)$$

### 2.5 Composite-Order Asymmetric Bilinear Groups and Subgroup Decision Assumptions

We will also consider generators that output composite-order bilinear groups, where the output $G := (N, G_1, G_2, G_T, e)$ features groups $G_1, G_2, G_T$ of a composite order $N = pq$, where $p$ and $q$ are primes of $\Theta(\lambda)$ bits. In this case, we will refer to use $g_p, g_q$ to denote the individual generators of the subgroups of $G_1$ of prime orders $p$ and $q$ respectively. We will similarly use $h_p, h_q$ to denote the generators of the subgroups of $G_2$ of prime orders $p$ and $q$.

Subgroup decision assumptions state that it is hard to distinguish a random element of a subgroup of a composite-order bilinear group from a random element of the full group, so long as the adversary is not allowed access to an element of the missing subgroup on the opposite side (since this would allow a trivial distinguisher via the pairing operation). We now define two such assumptions:

**Definition 4** (Subgroup Decision Assumption 1). We say that Subgroup Decision Assumption 1 holds if for all PPT adversaries $A$, the following advantage function is
negligible in $\lambda$.

$$\text{Adv}^{\text{SD1}}_A(\lambda) := |\Pr[\mathcal{A}(N, g_p, g_q, h_p, g_p^s) = 1] - \Pr[\mathcal{A}(N, g_p, g_q, h_p, g_p^s g_q^\tilde{s}) = 1]|$$

where $\mathbb{G} := (N, G_1, G_2, G_T, e) \leftarrow \mathcal{G}$, $(g_p, g_q), (h_p, h_q)$ are generators of the subgroups of $G_1, G_2$ of prime-orders $p, q$ where $N = pq$, and $s, \tilde{s} \leftarrow \mathbb{Z}_p$

**Definition 5** (Subgroup Decision Assumption 2). We say that Subgroup Decision Assumption 2 holds if for all PPT adversaries $\mathcal{A}$, the following advantage function is negligible in $\lambda$.

$$\text{Adv}^{\text{SD2}}_A(\lambda) := |\Pr[\mathcal{A}(N, g_p, g_p^s g_q^\tilde{s}, h_p, h_p^r) = 1] - \Pr[\mathcal{A}(N, g_p, g_p^s g_q^\tilde{s}, h_p, h_p^r h_q^\tilde{r}) = 1]|$$

where $\mathbb{G} := (N, G_1, G_2, G_T, e) \leftarrow \mathcal{G}$, $(g_p, g_q), (h_p, h_q)$ are generators of the subgroups of $G_1, G_2$ of prime-orders $p, q$ where $N = pq$, and $r, \tilde{r}, s, \tilde{s} \leftarrow \mathbb{Z}_p$

Note that in Subgroup Decision Assumption 2, the adversary is allowed access to an element of the missing subgroup on the opposite side in the form of $g_p^s g_q^\tilde{s}$, but this restricted access (the $g_q$ component is tied with a random $g_p$ component) prevents the trivial pairing attack.
Chapter 3

Unbounded ABE

The state-of-the-art in terms of asymptotic efficiency for adaptively secure ABE for Boolean formulas from static assumptions is the construction of [Lew+10]. In this section, we exhibit the construction, describe the main efficiency issue that arises when supporting access policies beyond read-once Boolean formulas, and finally describe our contributions towards mitigating this issue by reducing the public parameter size to a constant.

3.1 Textbook ABE Construction

We now describe a KP-ABE construction in the spirit of [Lew+10; LW12] which is adaptively secure from (static) subgroup decision assumptions in composite-order bilinear groups, and is constructed entirely within a prime-order subgroup of a composite-order group of order $N = pq$. Here, ciphertexts $ct_x$ are associated with attribute vectors $x \in \{0, 1\}^n$ and keys $sk_f$ with Boolean formulas $f$ whose input nodes are

---

1. Recall from Section 2.5 that in this composite-order setting we use $g_p, g_q$ and $h_p, h_q$ to denote the generators of the subgroups of $G_1, G_2$ of prime orders $p, q$ and note that we can therefore express any element of $(G_1, G_2)$ as $(g_p^a g_q^b, h_p^c, h_q^d)$. The second prime-order subgroup is used only in the proof of security, so our construction will contain only $g_p, h_p$ raised to various exponents.

2. Some works associate ciphertexts with a set $S \subseteq [n]$ where $[n]$ is referred to as the attribute universe, in which case $x \in \{0, 1\}^n$ corresponds to the characteristic vector of $S$. 
mapped to indices in \([n]\) by a function \(\rho : [m] \rightarrow [n]\). Note that the \texttt{KeyGen} algorithm uses the \texttt{share} procedure from Section 2.2:

\section*{Setup}(1^\lambda, 1^n) : Run \(G = (N, G_1, G_2, G_T, e) \leftarrow G(1^\lambda)\). Sample \(\mu, w_i \leftarrow \mathbb{Z}_N\) (\(\forall i \in [n]\)) and output:

\[msk := (v, w_1, \ldots, w_n)\]
\[mpk := (g_p, g_p^{u_1}, \ldots, g_p^{u_n}, e(g_p, h_p)^v)\]

\section*{Enc} \((mpk, x, M) : \) Sample \(s \leftarrow \mathbb{Z}_N\). Output:

\[ct_x := (ct_1, \{ct_{2,i}\}_{x_i=1}^n, ct_3)\]
\[:= (g_p^s, \{g_p^{sw_i}\}_{x_i=1}^n, M \cdot e(g_p, h_p)^{sv})\]

\section*{KeyGen} \((mpk, msk, f) : \) Sample \(\{v_j\}_{j \in m'}\), \(\rho \leftarrow \text{share}(f, v), r_j \leftarrow \mathbb{Z}_{p_1}\). Output:

\[sk_f := \{sk_{1,j}, sk_{2,j}\}_{j \in \mathbb{m'}}\]
\[:= \{h_p^{v_j + r_j w_i(j)}, h_p^{v_j}\}_{j \in \mathbb{m'}}, \rho : [m'] \rightarrow [n]\]

\section*{Dec} \((mpk, sk_f, ct_x) : \) Compute \(c_j\) such that \(v = \sum_{x_{\rho(j)}=1} c_j v_j\) as described in Section 2.2. Output:

\[ct_3 \cdot \prod_{x_{\rho(j)}=1} \left(\frac{e(ct_{2,r(j)}, sk_{2,j})}{e(ct_1, sk_{1,j})}\right)^{c_j}\]

22
Correctness

Decryption works by computing \( \{e(g_p, h_p)^{s_{uv}}\}_{x_i=1} \), from which we can compute the blinding factor \( e(g_p, h_p)^{s_{uv}} \) via linear reconstruction of the secret sharing scheme “in the exponent”. Formally, correctness relies on the fact that for all \( j \), we have

\[
\frac{e(ct_1, sk_{1,j})}{e(ct_{2,\rho(j)}, sk_{2,j})} = e(g_p, h_p)^{s_{uv}}
\]

Therefore, for all \( f, x \) such that \( f(x) = 1 \), we have:

\[
ct_3 \cdot \prod_{x_{\rho(j)=1}} \left( \frac{e(ct_{2,\rho(j)}, sk_{2,j})}{e(ct_1, sk_{1,j})} \right)^{c_j} = M \cdot e(g_p, h_p)^{s_{uv}} \cdot \prod_{x_{\rho(j)=1}} e(g_p, h_p)^{-sc_j v_j}
\]

\[
= M \cdot e(g_p, h_p)^{s_{uv}} \cdot e(g_p, h_p)^{-s} \sum_{x_{\rho(j)=1}} \ c_j v_j
\]

\[
= M \cdot e(g_p, h_p)^{s_{uv}} \cdot e(g_p, h_p)^{-s}
\]

\[
= M
\]

Adaptive Security

The adaptive security of a construction in this vein was proven in [Lew+10] only in the “one-use setting,” where for each key, each attribute value \( w_i \) is used at most once (that is, map \( \rho \) is required to be injective). Obviously, this is a severe restriction on the class of supported access policies (“read-once Boolean formulas”), but the authors of [Lew+10] give a “meta-attribute” re-encoding technique which allows the construction to extend past the one-attribute-use setting (at the cost of compactness).
3.2 Meta-Attribute Encoding Technique

We now review the “meta-attribute” re-encoding technique described in [Lew+10] as a way to extend the construction to support Boolean formulas with attribute re-use. Suppose we have a KP-ABE system that is secure when each attribute is used at most once in Boolean formulas associated to its keys, but we would like to support access policies beyond this formula class: formulas where attributes can be used up to \( k \) times. The “meta-attribute” technique accommodates this by making \( k \) copies of each attribute. For each original attribute \( w_i \), we will have new “meta-attributest” \( w_{i,1}, \ldots, w_{i,k} \). Each time we want to re-use an original attribute \( w_i \) in a formula, we will use meta-attribute \( w_{i,j} \) for a different value of \( j \). Each time we want to associate attributes \( \mathbf{x} \) to a ciphertext, we instead use \( \{w_{i,1}, \ldots, w_{i,k}\}_{x_i=} \). Each meta-attribute is therefore used once, so we can rely on the one-use security of the original system, yet obtain the functionality afforded by policies with up to \( k \) uses.

We note that when this transformation is applied to the constructions of [Lew+10] and its lineage the size of the ciphertext and public parameters grow linearly with \( k \). This dependence is undesirable, since one can imagine complex access policies where attributes are re-used many times. Supporting such policies would require a hit to the construction’s efficiency.

A first attempt at mitigating this efficiency hit would be to reduce the public parameter dependence. In particular, we would like ABE schemes where the public parameters are of constant size. We call schemes with this property *unbounded* ABE. [LW11b] constructed the first unbounded attribute-based encryption scheme through
a slight modification to the classic [Lew+10] construction, but only achieved selective
security. [OT12] later constructed an adaptively secure unbounded ABE scheme
using seemingly different techniques. A natural question remained: can we view the
techniques of [OT12] in relation to the lineage of [Lew+10] and related frameworks?

3.3 Our Contribution

We now describe our contributions towards this problem that result in a more modu-
lar understanding of unbounded adaptively secure ABE from static assumptions: the
works of [KL15; Che+18]. Taken together, they show how to prove adaptive security
of the modularly constructed scheme of [LW11b], under the general $k$-Linear assump-
tion in bilinear groups. As a result of this enhanced understanding, we obtain an
adaptively-secure unbounded ABE construction that is more efficient than [OT12],
as well as the first unbounded ABE construction for arithmetic span programs.

The core of our contributions are two “entropy expansion lemmas,” which are
used to boost the entropy of a small number of public parameter group elements into
a large number of group elements. Our constructions can be viewed then as taking
our textbook ABE construction (derived from [Lew+10]) and applying a substitution
to the set of group elements in the public parameter that grows linearly with the
number of attributes. Each (random) group element is replaced with a function of
some smaller number of group elements. Our entropy expansion lemmas then allow
us to argue that this substitution is indistinguishable from the original case, when
each group element was uniformly random. At that point, the original (adaptive)
proof of [Lew+10] can go through. Since our entropy expansion lemmas rely only on static assumptions, we obtain an unbounded version of [Lew+10] that is adaptively secure under static assumptions.

**Textbook Security Proof**

The proof of [Lew+10] proceeds through what is now a classic application of the “dual system proof methodology” [Wat09]. A dual system encryption scheme is constructed so that an adversary cannot distinguish the distribution of normal keys (or ciphertexts) from special “semi-functional” keys (or ciphertexts). Semi-functional keys are capable of decrypting normal ciphertexts, but semi-functional keys cannot decrypt a semi-functional ciphertext. A typical dual system proof consists of a hybrid where the first step changes the construction of the challenge ciphertext: from normal to semi-functional. The hybrid then runs over each key requested by the adversary, replacing each requested key with a semi-functional key. At the end, only semi-functional keys are given to an adversary whose job is to break the security of a semi-functional ciphertext. Due to the way semi-functional ciphertexts and secret keys are constructed, it is typically easy to argue the game’s security at this point (semi-functional secret keys cannot be used to decrypt semi-functional ciphertexts, including the semi-functional challenge ciphertext).

**Semifunctional Distributions:**

We now describe the semifunctional ciphertext and secret key distributions used in the security proof of our textbook construction:
A ciphertext can be in one of the following forms:

- **Normal**: generated as in the scheme of Section 3.1:

  \[ \text{ct}_x = (g_p^s, \{g_p^{sw_i}\}_{x_i=1}, M \cdot e(g_p, h_p)^{sw}) \]

- **SF**: same as a **Normal** ciphertext, except we additionally draw \( \tilde{s} \leftarrow \mathbb{Z}_N \) and form:

  \[ \text{ct}_x = (g_p^s \tilde{g}_q^\tilde{s}, \{g_p^{sw_i} g_q^{\tilde{sw}_i}\}_{x_i=1}, M \cdot e(g_p, h_p)^{sw}) \]

A secret key can be in one of the following forms:

- **Normal**: generated as in the scheme of Section 3.1:

  \[ \text{sk}_f = \{h_p^{v_j+r_j w_{\rho(j)}}, h_p^{\tilde{r}_j w_{\rho(j)}}\}_{j \in [m'_1]}, \rho : [m'] \rightarrow [n] \]

- **P-Normal**: same as a **Normal** key, except we additionally draw \( \tilde{r}_j \leftarrow \mathbb{Z}_{p_2} \), compute \( (\{\tilde{v}_j\}_{j \in [m'_1]}, \rho) \leftarrow \text{share}(f, 0) \), and form:

  \[ \text{sk}_f = \{h_p^{v_j+r_j w_{\rho(j)}} h_q^{\tilde{v}_j+r_j w_{\rho(j)}}, h_p^{\tilde{r}_j h_q^{\tilde{r}_j}}\}_{j \in [m'_1]}, \rho : [m'] \rightarrow [n] \]

- **P-SF**: same as a **P-Normal** key, except we compute \( (\{\tilde{v}_j\}_{j \in [m'_1]}, \rho) \leftarrow \text{share}(f, \tilde{v}) \) for \( \tilde{v} \leftarrow \mathbb{Z}_{n} \), and form:

  \[ \text{sk}_f = \{h_p^{v_j+r_j w_{\rho(j)}} h_q^{\tilde{v}_j+r_j w_{\rho(j)}}, h_p^{\tilde{r}_j h_q^{\tilde{r}_j}}\}_{j \in [m'_1]}, \rho : [m'] \rightarrow [n] \]
• **SF**: same as a P-SF key, except we form:

\[ \text{sk}_j = \{ h_p^{x_j + r_j w_{p(j)}}, h_q^{r_j}, h_{p'}^{r_j} \}_{j \in [m']} \}, \rho : [m'] \rightarrow [n] \]

Here, P stands for pseudo following [Wee14; CGW15]. Note that while the normal construction is contained entirely within the \( p_1 \) subgroup, the semifunctional distributions feature a “dual” version of the original scheme in the \( p_2 \) subgroup, with completely independent randomness (the \( w_i \) in the \( p_2 \) subgroup are distributed equivalently to fresh \( \tilde{w}_i \) due to the Chinese Remainder Theorem), particularly, independent from the \( w_i \) in the public parameters.

**Hybrid Sequence:**

Suppose the adversary A makes at most \( Q \) secret key queries. The hybrid sequence is as follows:

- **H\(_0\)**: real game
- **H\(_1\)**: same as \( H_0 \), except we use a SF ciphertext.
- **H\(_{2,\ell,1}\)**, \( \ell = 0, \ldots, Q \): same as \( H_1 \), except the \( \ell \)'th key is P-Normal, the first \( \ell - 1 \) keys are SF and the last \( Q - \ell \) keys are Normal.
- **H\(_{2,\ell,2}\)**: same as \( H_{2,\ell,1} \) except the \( \ell \)'th key is P-SF.
- **H\(_{2,\ell,3}\)**: same as \( H_{2,\ell,1} \) except the \( \ell \)'th key is SF.
- **H\(_3\)**: replace \( M \) with random.
Proof Overview:

The dual system proof uses Subgroup Decision Assumptions \(^3\) to establish this dual component one key at a time (changing Normal to P-Normal) where we now have the pair:

\[
\begin{align*}
ct'_x &:= (g_q^i, \{g_q^{\tilde{w}_i}\}_{x_i=1}) \\
\sk'_f &:= (\{h_q^{\tilde{v}_j + \tilde{w}_{\mu(j)}}, h_q^{\tilde{r}_j}\}_{j \in [m']})
\end{align*}
\]

isolated as the only elements in the \(p_2\) subgroup. The security of the secret sharing scheme and the entropy in the \(p_2\) component \(\tilde{w}_i\) is used to change the key from P-Normal to P-SF, with one final Subgroup Decision Assumption to move the key from P-SF to SF, hiding the \(\tilde{w}_i\) and allowing them to be reused in later semifunctional keys while saving the progress of the secret shared \(\tilde{v}\) in the \(p_2\) component for this key.

In more detail,

- We have \(H_0 \approx_c H_1 \equiv H_{2,0,3}\) via Subgroup Decision Assumption 1, which tells us \((g_p, g_p^s) \approx_c (g_p, g_p^s g_q^s)\). Here, the security reduction will pick \(w_1, \ldots, w_n\) and \(v\) so that it can simulate the mpk, the ciphertext and the secret keys.

- We have \(H_{2,\ell-1,3} \approx_c H_{2,\ell,1}\) for all \(\ell \in [Q]\). The difference between the two is that we switch the \(\ell\)’th \(\sk_f\) from Normal to P-Normal. This follows from Subgroup Decision Assumption 2, which tells us \((h_p, h_p^s) \approx_c (h_p, h_p^s h_q^s)\) Again,

---

\(^3\)Recall from Section 2.5 that subgroup decision assumptions state that it is hard to distinguish a random element of a subgroup of a composite-order bilinear group from a random element of the full group, so long as the adversary is not allowed access to an element of the missing subgroup on the opposite side (since this would allow a trivial distinguisher via the pairing operation).
the security reduction will pick \(w_1, \ldots, w_n\) and \(v\) so that it can simulate the \(\text{mpk}\), the ciphertext and the secret keys (note that here, we must rerandomize the subgroup decision challenge to produce \(h_p^{r_j}\) for each \(j\)).

- We have \(H_{2,\ell,1} \approx_c H_{2,\ell,2}\), for all \(\ell \in [Q]\). The difference between the two is that we switch the \(\ell\)'th \(\text{sk}_f\) from \text{Normal} to \text{SF}. This is a key step in the proof where we rely on the security of the secret sharing scheme, enabled by the fact that all \(v_j\) for \(x_{\rho(j)} = 0\) are hidden by corresponding \(\bar{w}_i\) which are only used once (due to the one-use restriction) to hide each such \(v_j\) and do not occur in the public parameters or the challenge ciphertext (since \(x_{\rho(j)} = 0\)).

- We have \(H_{2,\ell,2} \approx_c H_{2,\ell,3}\) for all \(\ell \in [Q]\). The difference between the two is that we switch the \(\ell\)'th \(\text{sk}_f\) from \(\text{P-SF}\) to \(\text{SF}\).

This follows again via Subgroup Decision Assumption 2, which tells us \((h_p, h_p^r) \approx_c (h_p, h_p^r h_q^r)\), symmetrically to the proof for \(H_{2,\ell-1,3} \approx_c H_{2,\ell,1}\), except that shares of \(\bar{v}\) is used instead of shares of 0 in the \(\ell\)th secret key.

- We have \(H_{2,Q,3} \equiv H_3\). In \(H_{2,Q,3}\), the secret keys only leak \(v + \bar{v}_1, \ldots, v + \bar{v}_Q\). This means that \(e(g_p, g_q)^{sv}\) is statistically random.

### 3.4 Entropy Expansion Lemma

In [KL15], we attempt to mitigate the public parameter blowup described in Section 3.2 by “compressing” \(g_p^{w_1}, \ldots, g_p^{w_n}\) in \(\text{mpk}\). The main idea is to generate \(\{w_j\}_{j \in [n]}\) using subset sums of a logarithmically smaller set of elements \(\{a_k\}_{k \in [g \cdot n]}\), inspired by similarly “compressing” PRF constructions like Naor-Reingold [NR97] and its
2-Lin-based extension [LW09]. Simply replacing \( w_j \) with \( \sum_{j_k=1}^s a_k \) leads to natural malleability attacks on the ciphertext, and instead, we replace \( sw_j \) with \( s_j \sum_{j_k=1}^s a_k \), where \( s_1, \ldots, s_n \) are fresh randomness used in encryption. Next, we need to bind the \( s_j \sum_{j_k=1}^s a_k \)'s together via some common randomness \( s \); it suffices to use \( sw + s_j \sum_{j_k=1}^s a_k \) in the ciphertext. That is, we start with the scheme in Section 3.1 and we perform the substitutions (*):

\[
\text{ciphertext:} \quad (s, sw_i) \mapsto (s, sw + s_i \sum_{j_k=1}^s a_k, s_i) \quad (*)
\]

\[
\text{secret key:} \quad (v_j + w_{p(j)}r_j, r_j) \mapsto (v_j + r_j w, r_j \sum_{j_k=1}^s a_k)
\]

This yields the following scheme:

\[
\begin{align*}
\text{mpk} & := (g_p, g_p^w, g_p^{a_1}, \ldots, g_p^{a_n}, e(g_p, h_p)^v) \\
\text{ctx} & := (g_p^s, \{g_p^{i_k=1} \cdot g_p^{s_i}\}_{x_i=1}^M, \cdot e(g_p, h_p)^{sv}) \\
\text{skf} & := (\{h_p^{v_j + r_j w}, h_p^{r_j}, h_p \sum_{j_k=1}^s a_k\}_{j \in [n]})
\end{align*}
\]  

(3.2)

As a sanity check for decryption, observe that we can compute \( \{e(g_p, h_p)^{sv_j}\}_{x_j=1}^M \) and then \( e(g_p, h_p)^{sv} \) as before.

**Proof Strategy:**

The analysis of our scheme in (3.2) follows a simple and natural proof strategy: we move to a new subgroup, and try to “undo” the substitutions described in (*) to re-
cover ciphertext and keys similar to those in the Textbook KP-ABE from Section 3.1, upon which we can apply the same analysis as Section 3.3. Our [KL15] Entropy Expansion Lemma exactly ⁴ enables this:


\[
(g_p, g_p^w, g_p^{a_1}, \ldots, g_p^{a_{k_n}}, h_p, h_p^w, h_p^{a_1}, \ldots, h_p^{a_{k_n}}),
\]

we show that

\[
\left\{ \begin{array}{l}
g_p^s, \{g_p^{sw+j} : j \in \mathbb{N}\} \\
\{h_p^{r_j w}, h_p^{r_j}, h_p^{a_k} : j \in \mathbb{N}\}
\end{array} \right\} \approx_c \left\{ \begin{array}{l}
g_q^s, \{g_q^{sw+j} : j \in \mathbb{N}\} \\
\{h_q^{r_j w}, h_q^{r_j}, h_q^{a_k} : j \in \mathbb{N}\}
\end{array} \right\}
\]

where "—" is short-hand for duplicating the terms on the LHS, so that the \(g_p, h_p\)-components remain unchanged.

That is, starting with the LHS, we replaced (i) \(\sum_{j=1}^{k_n} a_k\) with fresh \(u_j\), and (ii) \(w\) with fresh \(w_j\), both in the \(p_2\)-subgroup.

Let us start with the simpler setting where the adversary makes only a single key query. Upon applying our entropy expansion lemma (and a subgroup decision assumption to introduce the \(h_q^{w_j}\)’s.), we have that the ciphertext/key pair \((ct_x, sk_f)\)

---

⁴Well, not exactly. The ABE construction and Entropy Expansion Lemma in [KL15] are proved secure in *symmetric* composite-order bilinear groups. For consistency of notation and ease of exposition, we instead cast the Lemma into the asymmetric setting to match our other results. As a consequence, the following lemma contains the “spirit” of [KL15], but is not actually proved in that paper. We tolerate this imprecision because the Lemma is strictly improved upon by [Che+18] as described later in Section 3.5, so understanding the precise details behind this weaker result would only be a distraction.
satisfies
\[
\begin{align*}
\left\{ g_p^{s+w+s_i} \sum_{i_k=1}^{s} a_k, g_p^{s_i} \right\}_{x_i=1} \\
\left\{ h_p^{v_j+r_j w}, h_p^{r_j}, h_p^{\rho(j)=1} \right\}_{j \in [m']} \\
\end{align*}
\approx_c . 
\left\{ g_q^{s+w+s_i u_i}, g_q^{s_i} \right\}_{x_i=1} \\
\left\{ h_q^{v_j+r_j w_{\rho(j)}}, h_q^{r_j}, h_q^{r_u_{\rho(j)}}, h_q^{\rho(j)=1} \right\}_{j \in [m']}
\right\}
\]

with \( M \cdot c(g_p, h_p)^{sw} \) omitted. Note that the boxed term on the RHS is exactly the Textbook KP-ABE ciphertext/key pair in (3.1) over the \( p_2 \)-subgroup, once we strip away the terms involving \( u_j, s_j \) (and where each exponent \( s, w_j, r_j, w_j v_j \) is identically distributed to independently drawn \( \tilde{s}, \tilde{w}_j, \tilde{r}_j, \tilde{w}_j \tilde{v}_j \) due to the Chinese Remainder Theorem).

Finally, to handle the general setting where the ABE adversary makes \( Q \) key queries, we simply reuse the same argument in a hybrid, incurring a factor of \( Q \) security loss (which is polynomial in the security parameter). At this point, we can rely on the (adaptive) security for the Textbook KP-ABE outlined in Section 3.3 for the setting with a single challenge ciphertext and \( Q \) key queries.

**Proof of Lemma**

We now sketch a proof of our Entropy Expansion Lemma 6. The proof proceeds in two steps:

---

33
(i) We replace $\sum_{j_k=1} a_k$ with fresh $u_j$; that is,
\[
\begin{align*}
\left\{ g_{p^{j_k=1}} \right\}_{j \in [n]} & \approx_c \left\{ \left\{ g_{p^{j_k=1}}^{u_j}, g_{p^{j_k=1}}^{s_j} \right\}_{j \in [n]} \right\} \\
\left\{ \left\{ h_{p^{j_k=1}}^{r_j}, h_{p^{j_k=1}} \right\}_{j \in [n]} \right\} & \approx_c \left\{ \left\{ h_{p^{j_k=1}}^{r_j}, h_{p^{j_k=1}}^{r_j} \right\}_{j \in [n]} \right\}
\end{align*}
\]

where we suppressed the terms involving $w$; moreover, this holds even given $g_{p^w}, g_{p^w}^{a_{ig}}, \ldots, g_{p^w}^{a_{ig}}$.

The main idea is that the base case when $\lg n = 2$ (and there is therefore just two elements $a_1, a_2$) is a direct variant of SXDH. We can then inductively prove the case for $\ell$ elements based on the security of the case for $\ell - 1$ elements. Given an instance of the problem for $(\ell - 1)$ elements, we can double the instance and rerandomize, then add in our own $\ell$th element to the second half, to produce what is all subsets of $\ell$ elements from the original $(\ell - 1)$ if we are in the left-hand-size of the challenge, and what is computationally close (under SXDH) to $2^\ell$ random elements if we are in the right-hand-size of the challenge.

(ii) Next, we replace $w$ with fresh $w_j$; that is,
\[
\begin{align*}
\left\{ g_{q^{s_j}}^{s_{w_j} + s_{j_k} u_j}, g_{q^{s_j}} \right\}_{j \in [n]} & \approx_c \left\{ g_{q^{s_j}}^{s_{w_j} + s_{j_k} u_j}, g_{q^{s_j}} \right\}_{j \in [n]} \\
\left\{ h_{q^{s_j}}^{r_j}, h_{q^{s_j}}^{r_j} \right\}_{j \in [n]} & \approx_c \left\{ h_{q^{s_j}}^{r_j}, h_{q^{s_j}}^{r_j} \right\}_{j \in [n]}
\end{align*}
\]

Intuitively, this should follow from the DDH assumption in the $p_2$-subgroup, which says that $(h_{q^{s_j}}^{r_j}, h_{q^{s_j}}) \approx_c (h_{q^{s_j}}^{r_j}, h_{q^{s_j}})$. The actual proof is more delicate since $w$ also appears on the other side of the pairing as $g_{q^{s_j}}^{s_{w_j} + s_{j_k} u_j}$; fortunately, we
can treat $u_j$ as a one-time pad that masks $w$.

The full proof is shown based on subgroup decision assumptions and the 2-Lin assumption in composite-order symmetric bilinear groups in [KL15], but the end ABE result is subsumed by the followup work described in the next section.

### 3.5 Entropy Expansion Lemma, Revisited

In [Che+18], we again address the challenge of “compressing” $g_p^{w_1}, \ldots, g_p^{w_n}$ in mpk, this time down to a constant number of group elements. The main idea following [LW11b; OT12] is to generate $\{w_j\}_{j \in [n]}$ via a pairwise-independent hash function as $w_0 + j \cdot w_1$, as in the Lewko-Waters IBE. Simply replacing $w_j$ with $w_0 + j \cdot w_1$ leads to natural malleability attacks on the ciphertext, and instead, we would replace $sw_j$ with $s_j(w_0 + j \cdot w_1)$, where $s_1, \ldots, s_n$ are fresh randomness used in encryption. Again, we need to bind the $s_j(w_0 + j \cdot w_1)$’s together via some common randomness $s$; it suffices to use $sw + s_j(w_0 + j \cdot w_1)$ in the ciphertext. That is, we again start with the [Lew+10] scheme from Section 3.1 and perform the substitutions (**):

<table>
<thead>
<tr>
<th>Ciphertext</th>
<th>Secret Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(s, sw_i)$</td>
<td>$(v_j + w_{\rho(j)} r_j, r_j)$</td>
</tr>
<tr>
<td>$(s, sw + s_i(w_0 + i \cdot w_1), s_i)$</td>
<td>$(v_j + r_j w, r_j, r_j(w_0 + \rho(j) \cdot w_1))$</td>
</tr>
</tbody>
</table>

(* *)
This yields the following scheme:

\[
\begin{align*}
\text{mpk} & := (g_p, g_p^w, g_p^{w_0}, g_p^{w_1}, e(g_p, h_p)^v) \\
\text{ct}_x & := (g_p^s, \{g_p^{\sum_i (w_0+i \cdot w_1)}, g_p^{s_j}\}_{j=1}, M \cdot e(g_p, h_p)^s) \\
\text{sk}_f & := (\{h_p^{w_j+r_j w}, h_p^{r_j}, h_p^{r_j(w_0+\rho(j) \cdot w_1)}\}_{j \in [m']})
\end{align*}
\]

Note that we can again compute \(\{e(g_p, h_p)^{s_{w_j}}\}_{x_j=1}\) and then \(e(g_p, h_p)^s\). We remark that the ensuing scheme is similar to Attrapadung’s unbounded KP-ABE in [Att14](Section 7.1), except the latter requires \(q\)-type assumptions.\(^5\)

**Proof Strategy:**

The analysis of our scheme in (3.5) follows the same strategy as that of [KL15]: we move to a new subgroup, and try to “undo” the substitutions described in (***) to recover ciphertext and keys similar to those in the [Lew+10] KP-ABE, upon which we can apply the same analysis as Section 3.3. Our [Che+18] Entropy Expansion Lemma exactly enables this:

**Lemma 7** ([Che+18] Entropy Expansion). *Given public parameters* \((g_p, g_p^w, g_p^{w_0}, g_p^{w_1}, h_p, h_p^w, h_p^{w_0}, h_p^{w_1})\)

\(^5\)Attrapadung’s unbounded KP-ABE does have the advantage that there is no read-once restriction on the span programs, but even with the read-once restriction, the proof still requires \(q\)-type assumptions.
we show that

\[
\mathcal{g}_p^s, \{g_p^{s w + s_j (w_0 + j \cdot w_1)}, g_p^{s_j}\}_{j \in [n]} \approx_C \mathcal{g}_q^s, \{g_q^{s v_j + s_j u_j}, g_q^{s_j}\}_{j \in [n]}
\]

where "—" is short-hand for duplicating the terms on the LHS, so that the \(g_p, h_p\) -
components remain unchanged.

That is, starting with the LHS, we replaced (i) \(\sum_{j_k=1} a_k\) with fresh \(u_j\), and (ii) \(w\)
with fresh \(w_j\), both in the \(p_2\)-subgroup.

Notice that the form of this lemma is almost identical to that of Lemma 6 from
Section 3.4. Indeed, the right-hand side of new elements in the \(p_2\)-subgroup are identi-
tical! Unsurprisingly, we can use this entropy expansion lemma in a nearly identical
way to prove the security of the scheme in (3.5). One remark is that while [KL15] uses
a hybrid argument to move from the single key to \(Q\)-key case, incurring a security
loss of a factor of \(Q\), [Che+18] takes advantage of the random self-reducibility of the
lemma components, allowing it to translate to the \(Q\)-key case with no security loss.

Proof of Lemma:

We now provide a proof sketch of our Entropy Expansion Lemma 7:

As before, the proof proceeds in two steps: (i) replacing \(w_0 + j \cdot w_1\) with fresh \(u_j\),
and then (ii) replacing \(w\) with fresh \(v_j\).
• We replace $w_0 + j \cdot w_1$ with fresh $u_j$; that is,
\[
\left\{ g_p^{s_j(w_0 + j \cdot w_1)}, g_p^{s_j} \right\}_{j \in [n]} \approx_c . \left\{ g_q^{s_j u_j}, g_q^{s_j} \right\}_{j \in [n]} \tag{3.7}
\]
where we suppressed the terms involving $w$; moreover, this holds even given $g_p, g_p^{u_0}, g_p^{u_1}$. Our first observation is that we can easily adapt the proof of Lewko-Waters IBE [LW10; CW14] to show that for each $i \in [n],
\[
\left\{ g_p^{s_i(w_0 + i \cdot w_1)}, g_p^{s_i} \right\} \approx_c . \left\{ g_q^{s_i u_i}, g_q^{s_i} \right\}_{j \neq i} \tag{3.8}
\]
The idea is that the first term on the LHS corresponds to an encryption for the identity $i$, and the next $n - 1$ terms correspond to secret keys for identities $j \neq i$; on the right, we have the corresponding “semi-functional entities”. At this point, we can easily handle $(h_q^{r_j}, h_q^{r_j(w_0 + i \cdot w_1)})$ via a statistical argument, thanks to the entropy in $w_0 + i \cdot w_1 \mod p_2$. Next, we need to get from a single $(g_p^{s_i(w_0 + i \cdot w_1)}, g_p^{s_i})$ on the LHS in (3.8) to $n$ such terms on the LHS in (3.7). This requires a delicate “two slot” hybrid argument over $i \in [n]$ and the use of an additional subgroup; similar arguments also appeared in [OT12; HJK12]. This necessitates a composite order group with three prime-order subgroups, whereas the Lewko-Waters IBE and the statement in (3.8) works with two primes in the asymmetric setting.
• Next, we replace $w$ with fresh $v_j$; that is,

$$\begin{align*}
\begin{cases}
g_q^s, \{g_q^{s_{w_j} + g_{v_j}}, g_q^s\}_{j \in [n]} \\
\{h_q^{r_{w_j}}, h_q^{r_j}, h_q^{r_{v_j}}\}_{j \in [n]}
\end{cases}
\approx_c
\begin{cases}
g_q^s, \{g_q^{s_{v_j} + g_{v_j}}, g_q^s\}_{j \in [n]} \\
\{h_q^{r_{v_j}}, h_q^{r_j}, h_q^{r_{v_j}}\}_{j \in [n]}
\end{cases}
\end{align*}$$

This step is the same as that of Lemma 6 in Section 3.4.

The full proof is shown based on the $k$-Linear assumption in asymmetric bilinear groups in [Che+18].

### 3.6 Concluding Remarks

The works of [KL15] and [Che+18] described in the previous sections result in constructions for KP-ABE and CP-ABE that are unbounded. That is, the public parameters of the ABE schemes in [Che+18] are of a constant size. So, while these constructions still suffer from the one-use restriction of the [Lew+10] construction that their security reduces to and the corresponding blowup that results when adapting to the multi-use setting using the transformation described in Section 3.2, the effect of this blowup is now mitigated – in particular, it no longer affects the public parameters, and is only seen in the ciphertext size increase.

Furthermore, we note that the techniques of [KL15] and [Che+18] are modular, and allow us to see how unbounded ABE works in comparison to the classic scheme of [Lew+10]. This enhanced understanding results in shorter ciphertexts and secret keys than the only comparable unbounded scheme of [OT12], as well as extends the access policies supported from read-once Boolean formulas to the larger class of read-once
arithmetic span programs [IW14], which capture many natural computational models, such as Boolean and arithmetic branching programs. Prior to [Che+18], we did not even know any selectively secure, unbounded ABE for arithmetic span programs from a static assumption.
Chapter 4

Adaptive ABE

4.1 Overview

Our constructions of adaptively secure ABE for Boolean formulas leverage a simplification of the recent “partial selectivization” framework for adaptive security [Jaf+17] (which in turn builds upon [Fuc+14; FJP15; Hem+16; JW16]) along with the classic dual system encryption methodology [Wat09; LW12].

Our starting point is again the textbook framework for constructing compact adaptively secure ABE [LW12] which is based on the dual system encryption methodology [Wat09; LW10; Lew+10] and described previously in Section 3.1. Throughout, we focus on monotone NC¹ circuit access policies, and note that the constructions extend readily to the non-monotone setting¹. Let \((G_1, G_2, G_T)\) be an asymmetric bilinear group of prime order \(p\), where \(g, h\) are generators of \(G_1, G_2\) respectively.

Recall the textbook KP-ABE from Section 3.1. Here, ciphertexts \(\text{ct}_x\) are associ-

¹Most directly by pushing all NOT gates to the input nodes of each circuit and using new attributes to represent the negation of each original attribute. It is likely that the efficiency hit introduced by this transformation can be removed through more advanced techniques à la [OSW07; LSW10], but we leave this for future work.
ated with attribute vectors \( x \in \{0, 1\}^n \) and keys \( \text{sk}_f \) with Boolean formulas \( f \):

\[
\text{msk} := (\mu, w_1, \ldots, w_n) \tag{4.1}
\]

\[
\text{mpk} := (g, g^{w_1}, \ldots, g^{w_n}, e(g, h)^\mu),
\]

\[
\text{ct}_x := (g^s, \{g^{sw_i}\}_{x_i=1}^{m}, e(g, h)^{\mu s} \cdot M)
\]

\[
\text{sk}_f := (\{h^{\mu_j + p_jw_{\rho(j)}}, h^{\nu_j}\}_{j\in[m]}, \rho : [m] \rightarrow [n])
\]

where \( \mu_1, \ldots, \mu_m \) are shares of \( \mu \in \mathbb{Z}_p \) w.r.t. the formula \( f \); the shares satisfy the requirement that for any \( x \in \{0, 1\}^n \), the shares \( \{\mu_j\}_{x_{\rho(j)}=1} \) determine \( \mu \) if \( x \) satisfies \( f \) (i.e., \( f(x) = 1 \)), and reveal nothing about \( \mu \) otherwise; and \( \rho \) is a mapping from the indices of the shares (in \([m]\)) to the indices of the attributes (in \([n]\)) to which they are associated. For decryption, observe that we can compute \( \{e(g, h)^{\mu_j s}\}_{x_i=1} \), from which we can compute the blinding factor \( e(g, h)^{\mu s} \) via linear reconstruction “in the exponent”.

Here, \( m \) is polynomial in the formula size, and we should think of \( m = \text{poly}(n) \gg n \). Note that the ciphertext consists only of \( O(n) \) group elements and therefore satisfies our compactness requirement.

**Proving Adaptive Security**

The crux of the proof of adaptive security lies in proving that \( \mu \) remains computationally hidden given just a single ciphertext and a single key and no \( \text{mpk} \) (the more general setting with \( \text{mpk} \) and multiple keys follows via what is by now a textbook
application of the dual system encryption methodology, as discussed in Section 3.3).

In fact, it suffices to show that $\mu$ is hidden given just

$$
\text{ct}_x' := (\{w_i\}_{x_i=1}) \quad \text{// “stripped down” ct}_x
$$

$$
\text{sk}_f := (\{h^{\rho_j + \nu_j w_j(j)}, h^{r_j}\}_{j \in [n]})
$$

where $x, f$ are adaptively chosen subject to the constraint $f(x) = 0$. Henceforth, we refer to $(\text{ct}_x', \text{sk}_f)$ as our “core 1-ABE component.” Looking ahead to our formalization of adaptive security for this core 1-ABE, we actually require that $\mu$ is hidden even if the adversary sees $h^{w_1}, \ldots, h^{w_n}$; this turns out to be useful for the proof of our KP-ABE (for improved concrete efficiency).

Core Technical Contribution

The technical novelty of this work lies in proving adaptive security of the core 1-ABE component under the DDH assumption. Previous analysis either relies on a $q$-type assumption [LW12; BSW07; Att14; AC17], or imposes a “one-use restriction” (that is, $\rho$ is injective and $m = n$, in which case security can be achieved unconditionally) [Lew+10; Wee14]. Our analysis relies on a Piecewise Guessing framework which refines and simplifies a recent framework of Jafargholi et al. for proving adaptive security via pebbling games [Jaf+17] (which in turn builds upon [Fuc+14; FJP15; Hem+16; JW16]).

Let $G_0$ denote the view of the adversary $(\text{ct}_x', \text{sk}_f)$ in the real game, and $G_1$ denote the same thing except we replace $\{\mu_j\}$ in $\text{sk}_f$ with shares of a random value indepen-
dent of $\mu$. Our goal is to show that $G_0 \approx_c G_1$. First, let us define an additional family
of games $\{H^U\}$ parameterized by $U \subseteq [m]$: $H^U$ is the same as $G_0$ except we replace
$\{\mu_j : j \in U\}$ in $sk_f$ with uniformly random values. In particular, $H^\emptyset = G_0$.

We begin with the “selective” setting, where the adversary specifies $x$ at the start of the game. Suppose we can show that $G_0 \approx_c G_1$ in this simpler setting via a series of $L + 1$ hybrids of the form:

$$G_0 = H^{h_0(x)} \approx_c H^{h_1(x)} \approx_c \cdots \approx_c H^{h_L(x)} = G_1$$

where $h_0, \ldots, h_L : \{0, 1\}^n \to \{U \subseteq [m] : |U| \leq R'\}$ are functions of the adversary’s choices $x$. Then, the Piecewise Guessing framework basically tells us that $G_0 \approx_c G_1$
in the adaptive setting with a security loss roughly $m^{R'} \cdot L$, where the factor $L$ comes
from the hybrid argument and the factor $m^{R'}$ comes from guessing $h_i(x)$ (a subset of $[m]$ of size at most $R'$). Ideally, we would want $m^{R'} \ll 2^n$, where $2^n$ is what we achieve from guessing $x$ itself.

First, we describe a straightforward approach which achieves $L = 2$ and $R' = m$
implicit in [LW12] (but incurs a huge security loss $2^m \gg 2^n$) where

$$h_1(x) = \{j : x_{\rho(j)} = 0\}.$$  

That is, $H^{h_1(x)}$ is $G_0$ with $\mu_j$ in $sk_f$ replaced by fresh $\mu'_j \leftarrow \mathbb{Z}_p$ for all $j$ satisfying $x_{\rho(j)} = 0$. Here, we have

- $G_0 \approx_c H^{h_1(x)}$ via DDH, since $h^{\mu_j + u_{\rho(j)}r_j}, h^{r_j}$ computationally hides $\mu_j$ whenever
\( x_{\rho(j)} = 0 \) and \( w_{\rho(j)} \) is not leaked in \( \mathbf{ct}_x \):

- \( H^{h_1(x)} \approx_{s} G_1 \) via security of the secret-sharing scheme since the shares \( \{\mu_j : x_{\rho(j)} = 1\} \) leak no information about \( \mu \) whenever \( f(x) = 0. \)

This approach is completely generic and works for any secret-sharing scheme.

In our construction, we use a variant of the secret-sharing scheme for \( \text{NC}^1 \) in [Jaf+17] (which is in turn a variant of Yao’s secret-sharing scheme [Vin+03; IK02]), for which the authors also gave a hybrid argument achieving \( L = 8^d \) and \( R' = O(d \log m) \) where \( d \) is the depth of the formula; this achieves a security loss \( 2^{O(d \log m)} \).

Recall that the circuit complexity class \( \text{NC}^1 \) is captured by Boolean formulas of logarithmic depth and fan-in two, so the security loss here is quasi-polynomial in \( n \). We provide a more detailed analysis of the functions \( h_0, h_1, \ldots, h_L \) used in their scheme, and show that the subsets of size \( O(d) \) output by these functions can be described only \( O(d) \) bits instead of \( O(d \log m) \) bits. Roughly speaking, we show that the subsets are essentially determined by a path of length \( d \) from the output gate to an input gate, which can be described using \( O(d) \) bits since the gates have fan-in two. Putting everything together, this allows us to achieve adaptive security for the core 1-ABE component with a security loss \( 2^{O(d)} = \text{poly}(n) \).

**Our ABE scheme.**

To complete the overview, we sketch our final ABE scheme which is secure under the \( k \)-Linear Assumption in prime-order bilinear groups.

To obtain prime-order analogues of the composite-order examples, we rely on
the previous framework of Chen et al. [CGW15; Gon+16; BKP14] for simulating composite-order groups in prime-order ones. Let \((G_1, G_2, G_T)\) be a bilinear group of prime order \(p\). We start with the KP-ABE scheme in (4.1) and carry out the following substitutions:

\[
g^* \mapsto [s^\top A]_1, \ h^{r_j} \mapsto [r_j]_2, \ w_i \mapsto W_i \leftarrow \mathbb{Z}_p^{(k+1) \times k}, \ \mu \mapsto v \leftarrow \mathbb{Z}_p^{k+1} \tag{4.2}
\]

where

\[
A \leftarrow \mathbb{Z}_p^{k \times (k+1)}, \ s, r_j \leftarrow \mathbb{Z}_p^k,
\]

\(k\) corresponds to the \(k\)-Lin Assumption desired for security\(^2\), and \([\_]_1, [\_]_2\) correspond respectively to exponentiations in the prime-order groups \(G_1, G_2\). We note that the naive transformation following [CGW15] would have required \(W_i\) of dimensions at least \((k + 1) \times (k + 1)\); here, we incorporated optimizations from [Gon+16; BKP14]. This yields the following prime-order KP-ABE scheme for \(\text{NC}^1\):

\[
\text{msk} := (v, W_1, \ldots, W_n) \\
\text{mpk} := ([A]_1, [AW_1]_1, \ldots, [AW_n]_1, e([A]_1, [v]_2), e([s^\top A]_1, [v]_2) \cdot M) \\
\text{ct}_x := ([s^\top A]_1, \{[s^\top AW_j]_1\}_{x=1}^n, e([s^\top A]_1, [v]_2) \cdot M) \\
\text{sk}_j := (\{[v_j + W_{\rho(j)}r_j]_2, [r_j]_2\}_{j \in [m]})
\]

\(^2\text{e.g: } k = 1\) corresponds to security under the Symmetric External Diffie-Hellman Assumption (SXDH), and \(k = 2\) corresponds to security under the Decisional Linear Assumption (DLIN).
where $v_j$ is the $j$’th share of $v$. Decryption proceeds as before by first computing

$$\{e([s^\top A]_1, [v_j]_2)\}_{\rho(j)=0 \lor \rho(j)=1}$$

and relies on the associativity relations $A W_i \cdot r_j = A \cdot W_i r_j$ for all $i, j$ [CW13].

In the proof, in place of the DDH assumption which allows us to argue that $(h^{w_j} r_j, h^{r_j})$ is pseudorandom, we will rely on the fact that by the $k$-Lin assumption, we have

$$(A, A W_i, [W_i r_j]_2, [r_j]_2) \approx_c (A, A W_i, [W_i r_j + \delta_{ij} a^\perp]_2, [r_j]_2)$$

where $A \leftarrow \mathbb{Z}_p^{k \times (k+1)}$, $W_i \leftarrow \mathbb{Z}_p^{(k+1) \times 2k}$, $r_j \leftarrow \mathbb{Z}_p^{2k}$ and $a^\perp \in \mathbb{Z}_{p+1}^{k+1}$ satisfies $A \cdot a^\perp = 0$.

### 4.2 Piecewise Guessing Framework for Adaptive Security

We now refine the adaptive security framework of [Jaf+17], making some simplifications along the way to yield the Piecewise Guessing framework that will support our security proof. We use $\langle A, G \rangle$ to denote the output of an adversary $A$ in an interactive game $G$, and an adversary wins if the output is 1, so that the winning probability is denoted by $\Pr[\langle A, G \rangle = 1]$.

Suppose we have two adaptive games $G_0$ and $G_1$ which we would like to show to be indistinguishable. In both games, an adversary $A$ makes some adaptive choices
that define $z \in \{0, 1\}^R$. Informally, the Piecewise Guessing framework tells us that if we can show that $G_0, G_1$ are $\epsilon$-indistinguishable in the selective setting where all choices defining $z$ are committed to in advance via a series of $L + 1$ hybrids, where each hybrid depends only on at most $R' \ll R$ bits of information about $z$, then $G_0, G_1$ are $2^{2R'} \cdot L \cdot \epsilon$-indistinguishable in the adaptive setting.

**Overview.**

We begin with the selective setting where the adversary commits to $z = z^*$ in advance. Suppose we can show that $G_0 \approx_c G_1$ in this simpler setting via a series of $L + 1$ hybrids of the form:

$$G_0 = H^{h_0(z^*)} \approx_c H^{h_1(z^*)} \approx_c \cdots \approx_c H^{h_L(z^*)} = G_1$$

where $h_0, \ldots, h_L : \{0, 1\}^R \to \{0, 1\}^{R'}$ and \{H^u\}_{u \in \{0, 1\}^R}$ is a family of games where the messages sent to the adversary in $H^u$ depend on $u$. In particular, the $\ell$th hybrid only depends on $h_\ell(z^*)$ where $|h_\ell(z^*)| \ll |z^*|$. Next, we describe how to slightly strengthen this hybrid sequence so that we can deduce that $G_0 \approx_c G_1$ even for an adaptive choice of $z$. Note that \{H^u\}_{u \in \{0, 1\}^R}$ is now a family of adaptive games where $z$ is adaptively defined as the game progresses. We have two requirements:

---

3Informally, \{H^u\} describes the simulated games used in the security reduction, where the reduction guesses $R'$ bits of information described by $u$ about some choices $z$ made by the adversary; these $R'$ bits of information are described by $h_\ell(z)$ in the $\ell$th hybrid. In the $\ell$th hybrid, the reduction guesses a $u \in \{0, 1\}^{R'}$ and simulates the game according to $H^u$ and hopes that the adversary will pick an $z$ such that $h_\ell(z) = u$; note that the adversary is not required to pick such an $z$. One way to think of $H^u$ is that the reduction is committed to $u$, but the adversary can do whatever it wants.
The first, *end-point equivalence*, just says the two equivalences

\[ G_0 = H^{h_0(z^*)}, \; G_1 = H^{h_L(z^*)} \]

hold even in the adaptive setting, that is, even if the adversary’s behavior defines an \( z \) different from \( z^* \). In our instantiation, \( h_0 \) and \( h_L \) are constant functions, so this equivalence will be immediate.

The second, *neighbor indistinguishability*, basically says that for any \( \ell \in [L] \), we have

\[ H^{u_0} \approx_c H^{u_1}, \; \forall u_0, u_1 \in \{0, 1\}^{R'} \]

as long as the adversary chooses \( z \) such that

\[ h_{\ell-1}(z) = u_0 \land h_\ell(z) = u_1 \]

It is easy to see that this is a generalization of \( H^{h_{\ell-1}(z^*)} \approx_c H^{h_\ell(z^*)} \) if we require \( z = z^* \). To formalize this statement, we need to formalize the restriction on the adversary’s choice of \( z \) by having the game output 0 whenever the restriction is violated.

That is, we define a pair of “selective” games \( \tilde{H}_{\ell,0}(u_0, u_1), \tilde{H}_{\ell,1}(u_0, u_1) \) for any \( u_0, u_1 \in \{0, 1\}^{R'} \), where

\[ \tilde{H}_{\ell,b}(u_0, u_1) \] is the same as \( H^{u_b} \), except we replace the output with 0 whenever \( (h_{\ell-1}(z), h_\ell(z)) \neq (u_0, u_1) \).

That is, in both games, the adversary “commits” in advance to \( u_0, u_1 \). Proving indistinguishability here is easier because the reduction knows \( u_0, u_1 \) and only needs
to handle adaptive choices of $z$ such that $(h_{t-1}(z), h_t(z)) = (u_0, u_1)$.

**Adaptive security lemma.**

The next lemma tells us that the two requirements above implies that $G_0 \approx_e G_1$ with a security loss $2^{2R} \cdot L$ (stated in the contra-positive). In our applications, $2^{2R}$ and $L$ will be polynomial in the security parameter.

**Lemma 8** (adaptive security lemma). Fix $G_0, G_1$ along with $h_0, h_1, \ldots, h_L : \{0, 1\}^R \to \{0, 1\}^{2R}$ and $\{H^u\}_{u \in \{0, 1\}^{2R}}$ such that

$$\forall z^* \in \{0, 1\}^R : H^{h_0(z^*)} = G_0, \ H^{h_L(z^*)} = G_1$$

Suppose there exists an adversary $A$ such that

$$\Pr[\langle A, G_0 \rangle = 1] - \Pr[\langle A, G_1 \rangle = 1] \geq \epsilon$$

then there exists $\ell \in [L]$ and $u_0, u_1 \in \{0, 1\}^{2R}$ such that

$$\Pr[\langle A, \hat{H}_{\ell,0}(u_0, u_1) \rangle = 1] - \Pr[\langle A, \hat{H}_{\ell,1}(u_0, u_1) \rangle = 1] \geq \frac{\epsilon}{2^{2R} \cdot L}$$

This lemma is essentially a restatement of the main theorem of [Jaf+17] (Theorem 2); we defer a comparison to the end of this section.

**Proof.** For the proof, we need to define the game $H_\ell(z^*)$ for all $\ell = 0, 1, \ldots, L$ and all $z^* \in \{0, 1\}^R$
\( H_\ell(z^*) \) is the same as \( H_{h_\ell(z^*)} \), except we replace the output with 0 whenever \( z \neq z^* \).

Roughly speaking, in \( H_\ell(z^*) \), the adversary “commits” to making choices \( z = z^* \) in advance.

- **Step 1.** We begin the proof by using “random guessing” to deduce that

\[
\Pr_{z^* \leftarrow \{0,1\}^R} [\langle A, H_0(z^*) \rangle = 1] - \Pr_{z^* \leftarrow \{0,1\}^R} [\langle A, H_L(z^*) \rangle = 1] \geq \frac{\epsilon}{2R}
\]

This follows from the fact that \( H_{h_0(z)} = G_0, H_{h_L(z)} = G_1 \) which implies

\[
\Pr_{z^* \leftarrow \{0,1\}^R} [\langle A, H_0(z^*) \rangle = 1] = \frac{1}{2R} \Pr[\langle A, G_0 \rangle = 1]
\]
\[
\Pr_{z^* \leftarrow \{0,1\}^R} [\langle A, H_L(z^*) \rangle = 1] = \frac{1}{2R} \Pr[\langle A, G_1 \rangle = 1].
\]

- **Step 2.** Via a standard hybrid argument, we have that there exists \( \ell \) such that

\[
\Pr_{z^* \leftarrow \{0,1\}^R} [\langle A, H_{\ell - 1}(z^*) \rangle = 1] - \Pr_{z^* \leftarrow \{0,1\}^R} [\langle A, H_\ell(z^*) \rangle = 1] \geq \frac{\epsilon}{2RL}
\]

which implies that:

\[
\sum_{z' \in \{0,1\}^R} [\langle A, H_{\ell - 1}(z') \rangle = 1] - \sum_{z' \in \{0,1\}^R} [\langle A, H_\ell(z') \rangle = 1] \geq \frac{\epsilon}{L}
\]

- **Step 3.** Next, we relate \( \hat{H}_{\ell,0}, \hat{H}_{\ell,1} \) and \( H_{\ell - 1}, H_\ell \). First, we define the set

\[
\mathcal{U}_\ell := \{(h_{\ell - 1}(z'), h_\ell(z')) : z' \in \{0,1\}^R \} \subseteq \{0,1\}^{R'} \times \{0,1\}^{R'}, \ell \in [L]
\]
Observe that for all \((u_0, u_1) \in \mathcal{U}_\ell\), we have

\[
\Pr[\langle A, \hat{H}_{\ell,1}(u_0, u_1) \rangle = 1] = \sum_{z' : (h_{\ell-1}(z'), h_\ell(z')) = (u_0, u_1)} \Pr[\langle A, H_\ell(z') \rangle = 1]
\]

Then, we have

\[
\sum_{z' \in \{0, 1\}^R} \Pr[\langle A, H_\ell(z') \rangle = 1] = \sum_{(u_0, u_1) \in \mathcal{U}_\ell} \left( \sum_{z' : (h_{\ell-1}(z'), h_\ell(z')) = (u_0, u_1)} \Pr[\langle A, H_\ell(z') \rangle = 1] \right)
= \sum_{(u_0, u_1) \in \mathcal{U}_\ell} \Pr[\langle A, \hat{H}_{\ell,1}(u_0, u_1) \rangle = 1]
\]

By the same reasoning, we also have

\[
\sum_{z' \in \{0, 1\}^R} \Pr[\langle A, H_{\ell-1}(z') \rangle = 1] = \sum_{(u_0, u_1) \in \mathcal{U}_\ell} \Pr[\langle A, \hat{H}_{\ell,0}(u_0, u_1) \rangle = 1]
\]

This means that

\[
\sum_{(u_0, u_1) \in \mathcal{U}_\ell} \left( \Pr[\langle A, \hat{H}_{\ell,0}(u_0, u_1) \rangle = 1] - \Pr[\langle A, \hat{H}_{\ell,1}(u_0, u_1) \rangle = 1] \right)
= \sum_{z' \in \{0, 1\}^R} \Pr[\langle A, H_{\ell-1}(z') \rangle = 1] - \sum_{z' \in \{0, 1\}^R} \Pr[\langle A, H_\ell(z') \rangle = 1] \geq \frac{\epsilon}{L}
\]

where the last inequality follows from Step 2.

- **Step 4.** By an averaging argument, and using the fact that \(|\mathcal{U}_\ell| \leq 2^{2R'}\), there
exists \((u_0, u_1) \in \mathcal{U}_\ell\) such that

\[
\Pr[\langle A, \hat{H}_{\ell,0}(u_0, u_1) \rangle = 1] - \Pr[\langle A, \hat{H}_{\ell,1}(u_0, u_1) \rangle = 1] \geq \frac{\epsilon}{2^{2R'} L}
\]

This completes the proof. Note that \(2^{2R'}\) can be replaced by \(\max_{\ell} |\mathcal{U}_{\ell}|\). \qed

**Comparison with [Jaf+17].**

Our Piecewise Guessing framework makes explicit the game \(H^u\) which are described implicitly in the applications of the framework in [Jaf+17]. Starting from \(H^u\) and \(h_0, \ldots, h_L\), we can generically specify the intermediate games \(\hat{H}_{\ell,0}, \hat{H}_{\ell,1}\) as well as the games \(H_0, \ldots, H_L\) used in the proof of security. The framework of [Jaf+17] does the opposite: it starts with the games \(H_0, \ldots, H_L\), and the theorem statement assumes the existence of \(h_0, \ldots, h_L\) and \(\hat{H}_{\ell,0}, \hat{H}_{\ell,1}\) that are “consistent” with \(H_0, \ldots, H_L\) (as defined via a “selectivization” operation). We believe that starting from \(H^u\) and \(h_0, \ldots, h_L\) yields a simpler and clearer framework which enjoys the advantage of not having to additionally construct and analyze \(\hat{H}_{\ell,0}, \hat{H}_{\ell,1}\) and \(H_{\ell}\) in the applications.

Finally, we point out that the sets \(\mathcal{U}\) and \(\mathcal{W}\) in [Jaf+17](Theorem 2) corresponds to \(\mathcal{U}_\ell\) and \(\{0, 1\}^R\) over here (that is, we do obtain the same bounds), and the \(i\)'th function \(h_i\) corresponds to the \(\ell\)'th function \(h_{\ell-1} \circ h_{\ell}\) over here.

### 4.3 Pebbling Strategy for NC\(^1\)

We now define a pebbling strategy for \(\text{NC}^1\) which will be used to define the functions \(h_0, \ldots, h_L\) we’ll use in the Piecewise Guessing framework. Fix a formula \(f : \{0, 1\}^n \rightarrow\)
\{0,1\} of size \(m\) and an input \(x \in \{0,1\}^n\) for which \(f(x) = 0\). A pebbling strategy specifies a sequence of \(L\) subsets of \([m]\), corresponding to subsets of input nodes and gates in \(f\) that are pebbled. We refer to each subset in the sequence as a pebbling configuration and the \(i\)’th term in this sequence is the output of \(h_i(f, x)\) (where the combination of \(f, x\) correspond to the adaptive choices \(z\) made in our security game that will be later analyzed in the Piecewise Guessing framework).

Our pebbling strategy is essentially the same as that in [Jaf+17](Section 4); the main difference is that we provide a better bound on the size of the description of each pebbling configuration in Theorem 10.

**Pebbling Rules**

Fix a formula \(f : \{0,1\}^n \rightarrow \{0,1\}\) and an input \(x \in \{0,1\}^n\) for which \(f(x) = 0\). We are allowed to place or remove pebbles on input nodes and gates in \(f\), subject to some rules.

The goal of a pebbling strategy is to find a sequence of pebbling instructions that follow the rules and starting with the initial configuration (in which there are no pebbles at all), will end up in a configuration where only the root gate has a pebble. Intuitively, the rules say that we can place a pebble a node or a gate if we know that the out-going wire will be 0. More formally,

**Definition 9** (Pebbling Rules).

1. Can place or remove a pebble on any AND gate for which (at least) one input wire comes out of a node with a pebble on it.

2. Can place or remove a pebble on any OR gate for which all of the incoming
wires come out of nodes which have pebbles on them.

3. Can place or remove a pebble on any input node for which \( x_i = 0 \).

Given \((f, x)\), a pebbling strategy returns a sequence of pebbling instructions of the form PEBBLE \( g \) or unPEBBLE \( g \) for some gate \( g \), with the property that each successively applied instruction follows the pebbling rules in Definition 9.

**Pebbling Strategy**

Given an \( \text{NC}^1 \) formula \( f \) (recall Section 2.1) and an input \( x \) on which the formula evaluates to 0, consider the pebbling instruction sequence returned by the following recursive procedure, which maintains the invariant that the output wire evaluates to 0 for each gate that the procedure is called upon. The strategy is described in Figure 4.1 and begins by calling \textbf{Pebble}(\( f, x, g^* \)) on the root gate \( g^* \). We give an example in Figure 4.2.

Note that if this procedure is called on the root gate of a formula \( f \) with an input \( x \) such that \( f(x) = 0 \), then every AND gate on which the \textbf{Pebble}() procedure is called will have at least one child node with an output wire which evaluates to 0, and every OR gate on which the \textbf{Pebble}() procedure is called will have child nodes with output wires which both evaluate to 0. Furthermore, by inspection, \textbf{Pebble}(\( f, x, g^* \)) returns a sequence of pebbling instructions for the circuit that follows the rules in Definition 9.
Pebble($f, x, g$):

**Input**: A node $g$ of an NC$^1$ formula $f$ with children $g_L$ and $g_R$ along with input $x$ defining values along the wires of $f$.

1. (Base Case) If $g$ is an input node, Return “PEBBLE $g$”.
2. (Recursive Case) If $g = \text{OR}$, first call Pebble($f, x, g_L$) to get a list of operations $\Lambda_L$, then call Pebble($f, x, g_R$) to get a second list of operations $\Lambda_R$. Return $\Lambda_L \circ \Lambda_R \circ \text{"PEBBLE } g \text{" } \circ \text{Reverse}(\Lambda_R) \circ \text{Reverse}(\Lambda_L)$
3. (Recursive Case) If $g = \text{AND}$, call Pebble($f, x, \cdot$) on the first child gate whose output wire evaluates to 0 on input $x$ to get a list of operations $\Lambda$. Return $\Lambda \circ \text{"PEBBLE } g \text{" } \circ \text{Reverse}(\Lambda)$

Reverse($\Lambda$):

**Input**: A list of instructions of the form “PEBBLE $g$” or “unPEBBLE $g$” for a gate $g$.

1. Return the list $\Lambda$ in the reverse order, additionally changing each original “PEBBLE ” instruction to “unPEBBLE ” and each original “unPEBBLE ” instruction to “PEBBLE ”.

Figure 4.1: NC$^1$ formula pebbling strategy.

**Analysis.**

To be useful in the Piecewise Guessing framework, we would like for the sequence of pebbling instructions to have the property that each configuration formed by successive applications of the instructions in the sequence is as short to describe as possible (i.e., minimize the maximum representation size $R'$). One way to achieve this is to have, at any configuration along the way, as few pebbles as possible. An even more succinct representation can be obtained if we allow many pebbles but have a way to succinctly represent their location. Additionally, we would like to minimize the worst-case length, $L$, of any sequence produced. We achieve these two goals in the following theorem.

**Theorem 10** (pebbling NC$^1$). For every input $x \in \{0, 1\}^n$ and any monotone formula
Figure 4.2: Intermediate pebbling configurations on input $x = 001$. The thick black outline around a node corresponds to having a pebble on the node. Note that steps 10-17 correspond to “undoing” steps 1-8 so that at the end of step 17, there is exactly one pebble on the $\lor$ node leading to the output node.

\[ f \text{ of depth } d \text{ and fan-in two for which } f(x) = 0, \text{ there exists a sequence of } L(d) = 8^d \text{ pebbling instructions such that every intermediate pebbling configuration can be described using } R'(d) = 3d \text{ bits.} \]

**Proof.** Follows from the joint statements of Lemma 11 and Lemma 13 applied to the pebbling strategy in Figure 4.1.

Comparison with [Jaf+17].

Note that the strategy reproduced in Figure 4.1 is essentially the same as one analyzed by [Jaf+17], which argued that every configuration induced by the pebbling instruction sequence it produces can be described using $d(\log m + 2)$ bits, where $m$
is the number of wires in the formula. This follows from the fact that each such pebbling configuration has at most $d$ gates with pebbled children, and we can specify each such gate using $\log m$ bits and the pebble-status of its two children using an additional two bits. Our Lemma 13 analyzes the same pebbling strategy but achieves a more succinct representation by leveraging the fact that not all configurations of $d$ pebbled gates are possible due to the pebbling strategy used, so we don’t need the full generality allowed by $d \cdot \log m$ bits. Instead, Lemmas 12 and 13 show that every configuration produced follows a pattern that can be described using only $3d$ bits.

**Lemma 11** ([Jaf+17]). The pebbling strategy in Figure 4.1 called on the root gate $g^*$ for a formula $f$ of depth $d$ with assignment $x$ such that $f(x) = 0$, $Pebble(f, x, g^*)$, returns a sequence of instructions of length at most $L(d) \leq 8^d$.

This bound is a special case of that shown in [Jaf+17](Lemma 2) for fan-in two circuits.

**Proof.** This statement follows inductively on the depth of the formula on which $Pebble()$ is called.

For the base case, when $d = 0$ (and $Pebble$ has therefore been called on an input node) there is just one instruction returned, and:

$$1 \leq 8^{(0)}$$

When $Pebble()$ is called on a node at depth $d > 0$, the node is either an OR gate or an AND gate.

58
When `Pebble()` is called on an OR gate, using our inductive hypothesis for the instructions returned for the subformula of depth $d - 1$, notice that the number of instructions returned is:

$$L(d-1) + L(d-1) + 1 + L(d-1) + L(d-1) = 8^{(d-1)} + 8^{(d-1)} + 1 + 8^{(d-1)} + 8^{(d-1)} = 4 \cdot 8^{(d-1)} + 1 \leq 8^d$$

When `Pebble()` is called on an AND gate, using our inductive hypothesis for the instructions returned for the subformula of depth $d - 1$, notice that the number of instructions returned is:

$$L(d - 1) + 1 + L(d - 1) = 8^{(d-1)} + 1 + 8^{(d-1)} = 2 \cdot 8^{(d-1)} + 1 \leq 8^d$$

We note that the following lemma is new to this work and will be used to bound the representation size $R(d)$ of any configuration produced by application of the instructions output by the pebbling strategy.

**Lemma 12** (structure of pebbling configuration). *Every configuration induced by application of the instructions produced by the pebbling strategy in Figure 4.1 called on the root gate $g^*$ of a formula $f$ of depth $d$ with assignment $x$ such that $f(x) = 0$, $Pebble(f, x, g^*)$, has the following property for all gates $g$ in $f$ with children $g_L, g_R$:

If any node in the sub-tree rooted at $g_R$ is pebbled, then there exists at most one pebble on the sub-tree rooted at $g_L$, namely a pebble on $g_L$ itself.
Proof. Call a node “good” if it satisfies the property above. First, we make the following observation about the behavior of Reverse(): applying Reverse() to a list of instructions inducing a list of configurations for which all nodes are “good” produces a new list for which this is true. This holds since Reverse() does not change the configurations induced by a list of instructions, just the ordering (which is reversed). This follows from a simple proof by induction on the length of the input instruction list and the fact that for an input list of instructions parsed as $L_1 \circ L_2$ for two smaller-length lists, we can implement Reverse($L_1 \circ L_2$) as Reverse($L_2$) $\circ$ Reverse($L_1$).

We proceed with our original proof via induction on the depth of the formula upon which Pebble() is called.

Inductive Hypothesis: For formulas $f$ of depth $d - 1$ with root gate $g^*$ and assignment $x$ such that $f(x) = 0$, Pebble($f, x, g^*$) returns a sequence of instructions that induces a sequence of configurations that (1) end with a configuration where $g^*$ is the only pebbled node, and satisfies: (2) in every configuration all nodes are “good.”

Base Case: when Pebble($f, x, g^*$) is called on a formula of depth 0, the formula consists of just an input node $g^*$. The (single) returned instruction PEBBLE $g^*$ then satisfies that in both the initial and final configuration, the single node $g^*$ is good. Also, the sequence ends in the configuration where $g^*$ is the only pebbled node.

Inductive Step: when Pebble($f, x, g^*$) is called on formula of depth $d > 0$. Let $g^*_L, g^*_R$ denote the children of the root gate $g^*$ (either an AND or OR gate). Note that the sub-formulas $f_{g^*_L}$ and $f_{g^*_R}$ rooted at $g^*_L$ and $g^*_R$ have depth $d - 1$. We proceed via a case analysis:
If $g^*$ is an AND gate, then suppose the sequence of instructions returned is

$$\text{Pebble}(f_{g^*_R}, x, g^*_R) \circ \text{PEBBLE} \ g^* \circ \text{Reverse}(\text{Pebble}(f_{g^*_L}, x, g_R))$$

(The case with $g^*_L$ instead of $g^*_R$ is handled analogously, even simpler). Suppose $\text{Pebble}(f_{g^*_R}, x, g^*_R)$ (and thus $\text{Reverse}(\text{Pebble}(f_{g^*_R}, x, g^*_R))$ produces $L_0$ instructions. We proceed via a case analysis:

- Take any of the first $L_0$ configurations (starting from 0’th). Here, all pebbles are in the subformula rooted at $g^*_R$. We can then apply part (2) of the inductive hypothesis to the subformula $f_{g^*_R}$ rooted at $g^*_R$ (of depth $d - 1$) to deduce that property “good” holds for all nodes in $f_{g^*_R}$. All nodes in $f_{g^*_R}$ are unpebbled in all configurations, so they are automatically good. Lastly, the root gate $g^*$ has no pebbled nodes in the subformula rooted at $g_L$, so it is also good.

- For the $(L_0 + 1)$’th configuration reached after PEBBLE $g^*$, there are only two pebbles, one on $g^*$ (from the PEBBLE $g^*$ instruction) and another on $g^*_R$ (from part (1) of our inductive hypothesis applied to the (depth $d - 1$) subformula $f_{g^*_R}$). It is clear that all nodes in this configuration are good.

- For the last $L_0$ configurations, there is one pebble on $g^*$ and all remaining pebbles are in the subformula rooted at $g^*_R$. Clearly, $g^*$ is good. All nodes in $f_{g^*_L}$ are unpebbled in all configurations, so they are also good. Moreover, we can apply the inductive hypothesis to $f_{g^*_R}$ combined with our observation that Reverse preserves property (2) of this hypothesis to deduce that all nodes in the subformula are also good for all configurations.
Lastly, notice that since the last $L_0$ instructions undo the first $L_0$ instructions, the final configuration features a single pebble on $g^*$.

If $g^*$ is an OR gate, then the sequence of instructions returned is

\[ \text{Pebble}(f_{g_L^*}, x, g_{L_L}) \circ \text{Pebble}(f_{g_R^*}, x, g_{R_L}) \circ \text{PEBBLE} \; g^* \circ \text{Reverse}(\text{Pebble}(f_{g_R^*}, x, g_{R_R})) \circ \text{Reverse}(\text{Pebble}(f_{g_L^*}, x, g_{L_R})) \]

Suppose $\text{Pebble}(f_{g_R^*}, x, g_{R_R}^*), \text{Pebble}(f_{g_L^*}, x, g_{L_L}^*)$, and thus $\text{Reverse}(\text{Pebble}(f_{g_R^*}, x, g_{R_R}^*)), \text{Reverse}(\text{Pebble}(f_{g_L^*}, x, g_{L_L}^*))$, produces $L_0, L_1$ instructions. We proceed via a case analysis:

- Take any of the first $L_0$ configurations (starting from $0$’th). Here, all pebbles are in the subformula $f_{g_L^*}$ rooted at $g_{L_L}^*$. We can then apply part (2) of the inductive hypothesis to (depth $d - 1$) $f_{g_L^*}$ to deduce that property “good” holds for all nodes in $f_{g_L^*}$. All nodes in the subformula rooted at $g_{R_R}^*$, $f_{g_R^*}$, are unpebbled in all configurations, so they are automatically good. Lastly, the root gate $g^*$ has no pebbled nodes in the subformula rooted at $g_{R_R}^*$, so it is also good. Finally, by part (1) of this application of the inductive hypothesis, we know that $L_0$th configuration features a single pebble on $g_{L_L}^*$.

- Take any of the next $L_1$ configurations (starting from the $L_0$’th). Here, all pebbles are in the subformula rooted at $g_{R_R}^*$ except for the single pebble on $g_{L_R}^*$. We can then apply part (2) of the inductive hypothesis to (depth $d - 1$) $f_{g_R^*}$ (of depth $d - 1$) to deduce that property “good” holds for all nodes in $f_{g_R^*}$. All nodes in the subformula rooted at $g_{L_L}^*$ have no pebbles in their own subformulas, so they are automatically good. Lastly, the root gate $g^*$ may have
pebbled nodes in the subformula rooted at $g^*_R$ but the only pebbled node in the subformula rooted at $g^*_L$ is $g^*_L$ itself, so it is also good. Finally, we know that the $L_0 + L_1$th configuration features two pebbles: a pebble on $g^*_L$ (from the first $L_0$ instructions), and a pebble on $g^*_R$ (by part (1) of this application of the inductive hypothesis).

- For the $(L_0 + L_1 + 1)$th configuration reached after PEBBLE $g^*$, there are only three pebbles, one on $g^*$ (from the PEBBLE $g^*$ instruction), one on $g^*_L$ (from the first $L_0$ instructions), and another on $g^*_R$ (from the next $L_1$ instructions). It is clear that all nodes in this configuration are good.

- For the next $L_1$ configurations (reversing the instructions of the set of size $L_1$), there is one pebble on $g^*$, one pebble on $g^*_L$, and all remaining pebbles are in the subformula rooted at $g^*_R$, $f_{g^*_R}$. $g^*$ is good, since it only has one pebble in the subformula rooted at $g^*_L$, on $g^*_L$ itself. All nodes in the subformula rooted at $g^*_L$ have no pebbles in their own subformulas, so they are also good. Moreover, we can apply the inductive hypothesis to (depth $d - 1$) $f_{g^*_R}$ combined with our observation that Reverse preserves property (2) of this hypothesis to deduce that all nodes in $f_{g^*_R}$ are also good for all configurations. Note the final configuration in this sequence then contains two pebbles, one of $g^*$ and one on $g^*_L$.

- For the final $L_0$ configurations (reversing the instructions of the set of size $L_0$), there is one pebble on $g^*$, and all remaining pebbles are in the subformula rooted at $g^*_L$. $g^*$ is good, since it has no pebbles in the subformula rooted at $g^*_R$. Similarly, all nodes in the subformula rooted at $g^*_R$ are also good. Moreover, we can apply the inductive hypothesis to (depth $d - 1$) $f_{g^*_L}$ combined with our
observation that Reverse preserves property (2) of this hypothesis to deduce that all nodes in \( f_{g_l} \) are also good for all configurations.

Lastly, notice that since the last \( L_0 + L_1 \) instructions undo the first \( L_0 + L_1 \) instructions, the final configuration features a single pebble on \( g^* \).

\[
\square
\]

**Lemma 13** \((R'(d) = 3d)\). Every configuration induced by application of the instructions produced by the pebbling strategy in Figure 4.1 for a formula \( f \) of depth \( d \) with assignment \( x \) such that \( f(x) = 0 \) can be described using \( R'(d) = 3d \) bits.

*Proof.* We can interpret 3\( d \) bits in the following way to specify a pebbling: the first \( d \) bits specify a path down the formula starting at the root gate (moving left or right based on the setting of each bit), the next \( 2(d - 1) \) bits specify, for each of the \( (d - 1) \) non-input nodes along the path, which of its children are pebbled. Finally one of the last 2 bits is used to denote if the root node is pebbled.

From Lemma 12, we know that for all gates \( g \) with children \( g_L, g_R \), if any node in the sub-tree rooted at \( g_R \) is pebbled, then there exists at most one pebble on the sub-tree rooted at \( g_L \), namely a pebble on \( g_L \) itself. So, given a pebbling configuration, we can start at the root node and describe the path defined by taking the child with more pebbles on its subtree using \( d \) bits. All pebbles in the configuration are either on the root node or on children of nodes on this path and therefore describable in the remaining \( 2d \) bits.

\[
\square
\]
4.4 Core Adaptive Security Component

In this section, we will describe the secret-sharing scheme (\texttt{share}, \texttt{reconstruct}) used in our ABE construction. In addition, we describe a core component of our final ABE, and prove adaptive security using the pebbling strategy defined and analyzed in Section 4.3 to define hybrids in the Piecewise Guessing framework of Section 4.2.

Overview.

As described in the overview in Section 4.1, we will consider the following “core 1-ABE component”:

\[
ct'_x := (\{w_i\}_{x_i=1}) \quad \text{// “stripped down” ct}_x
\]

\[
\text{sk}_f := \{h_{\mu_j}\}_{\rho(j)=0} \cup \{h_{\mu_j+r_{\rho(j)}w_{\rho(j)}}, h'_{\rho(j)}\}_{\rho(j)\neq 0}
\]

where \((\{\mu_j\}, \rho) \leftarrow \text{share}(f, \mu)\).

We want to show that under the DDH assumption, \(\mu\) is hidden given just \((ct'_x, sk_f)\) where \(x, f\) are adaptively chosen subject to the constraint \(f(x) = 0\). We formalize this via a pair of games \(G_0^{\text{1-ABE}}, G_1^{\text{1-ABE}}\) and the requirement \(G_0^{\text{1-ABE}} \approx_c G_1^{\text{1-ABE}}\). In fact, we will study a more abstract construction based on any CPA-secure encryption with:

\[
ct'_x := (\{w_i\}_{x_i=1}) \quad \text{// “stripped down” ct}_x
\]

\[
\text{sk}_f' := \{\mu_j\}_{\rho(j)=0} \cup \{\text{CPA.Enc}(w_{\rho(j)}, \mu_j)\}_{\rho(j)\neq 0} \text{ where } (\{\mu_j\}, \rho) \leftarrow \text{share}(f, \mu)
\]
Linear secret sharing for $\text{NC}^1$

We first describe a linear secret-sharing scheme for $\text{NC}^1$; this is essentially the information-theoretic version of Yao’s secret-sharing for $\text{NC}^1$ in [Jaf+17; Vin+03; IK02]. It suffices to work with Boolean formulas where gates have fan-in 2 and fan-out 1, thanks to the transformation in Section 2.1. We describe the scheme in Figure 4.3, and give an example in Figure 4.4. Note that our non-standard definition of secret-sharing in Section 2.2 allows the setting of $\rho(j) = 0$ for shares that are available for reconstruction for all $x$. We remark that the output of share satisfies $|\{\mu_j\}| \leq 2m$ since each of

\begin{figure}
\begin{center}
\begin{tikzpicture}
\node[anchor=north west,inner sep=0] (image) at (0,0) {share$(f, \mu)$:
\begin{itemize}
\item Input: A formula $f : \{0,1\}^n \rightarrow \{0,1\}$ of size $m$ and a secret $\mu \in \mathbb{Z}_p$.
\item 1. For each non-output wire $j = 1, \ldots, m-1$, pick a uniformly random $\hat{\mu}_j \gets \mathbb{Z}_p$. For the output wire, set $\hat{\mu}_m = \mu$.
\item 2. For each outgoing wire $j$ from input node $i$, add $\mu_j = \hat{\mu}_j$ to the output set of shares and set $\rho(j) = i$.
\item 3. For each AND gate $g$ with input wires $a, b$ and output wire $c$, add $\mu_c = \hat{\mu}_c + \hat{\mu}_a + \hat{\mu}_b \in \mathbb{Z}_p$ to the output set of shares and set $\rho(c) = 0$.
\item 4. For each OR gate $g$ with input wires $a, b$ and output wire $c$, add $\mu_{ca} = \hat{\mu}_c + \hat{\mu}_a \in \mathbb{Z}_p$ and $\mu_{cb} = \hat{\mu}_c + \hat{\mu}_b \in \mathbb{Z}_p$ to the output set of shares and set $\rho(c_a) = 0$ and $\rho(c_b) = 0$.
\item 5. Output $\{\mu_j\}, \rho$.
\end{itemize}}
\end{tikzpicture}
\end{center}
\end{figure}

Figure 4.3: Information-theoretic linear secret sharing scheme $\text{share}$ for $\text{NC}^1$

the $m$ nodes adds a single $\mu_j$ to the output set, except for OR gates which add two: $\mu_{ja}$ and $\mu_{jb}$.

The reconstruction procedure $\text{reconstruct}$ of the scheme is essentially applying the appropriate linear operations to get the output wire value $\hat{\mu}_c$ at each node starting from the leaves of the formula to get to the root $\hat{\mu}_m = \mu$.

- Given $\hat{\mu}_a, \hat{\mu}_b$ associated with the input wires of an AND gate, we recover the
gate’s output wire value $\hat{\mu}_c$ by subtracting their values from $\mu_c$ (which is available since $\rho(c) = 0$).

- Given one of $\hat{\mu}_a, \hat{\mu}_b$ associated with the input wires of an OR gate, we recover the gate’s output wire value $\hat{\mu}_c$ by subtracting it from the appropriate choice of $\mu_{c_a}$ or $\mu_{c_b}$ (which are both available since $\rho(c_a) = \rho(c_b) = 0$).

Note that $\text{reconstruct}(f, x, \{\mu_j\}_{\rho(j)=0\lor x, \rho(j)=1})$ computes a linear operation with respect to the shares $\mu_j$. This follows from the fact that the operation at each gate in reconstruction is a linear operation, and the composition of linear operations is itself a linear operation. Therefore, $\text{reconstruct}(f, x, \{\mu_j\}_{\rho(j)=0\lor x, \rho(j)=1})$ is equivalent to identifying the coefficients $\omega_j$ of this linear function, where $\mu = \sum_{\rho(j)=0\lor x, \rho(j)=1} \omega_j \mu_j$.

As with any linear secret-sharing scheme, $\text{share}$ and $\text{reconstruct}$ can be extended in the natural way to accommodate vectors of secrets. Specifically, for a vector of
secrets \( v \in \mathbb{Z}_p^k \), define:

\[
\text{share}(f, v) := (\{ v_j := (v_{1,j}, \ldots, v_{k,j}) \}, \rho) \text{ where } (\{ v_i \}, \rho) \leftarrow \text{share}(f, v_i)
\]

(note that \( \rho \) is identical for all \( i \)). \text{reconstruct} can also be defined component-wise:

\[
\text{reconstruct}(f, x, \{ v_j \} \rho(j) = 0 \forall x \rho(j) = 1) := \sum_{\rho(j) = 0 \forall x \rho(j) = 1} \omega_j v_j \text{ where } \omega_j \text{ are computed as above}
\]

Our final ABE construction will use this extension.

**Core 1-ABE Security Game**

**Definition 14** (core 1-ABE security \( G_0^{1-ABE}, G_1^{1-ABE} \)). For a stateful adversary \( A \), we define the following games \( G_{\beta}^{1-ABE} \) for \( \beta \in \{0, 1\} \).

\[
\langle A, G_{\beta}^{1-ABE} \rangle := \{ \begin{array}{l}
\mu^{(0)}, \mu^{(1)} \leftarrow \mathbb{Z}_p; w_i \leftarrow \text{CPA.Setup}(\lambda) \\
b' \leftarrow A^{O_F(\cdot), O_X(\cdot), O_E(\cdot)}(\mu^{(0)}) \\
\end{array} \}
\]

where the adversary \( A \) adaptively interacts with three oracles:

\[
O_F(f) := \{ sk_f = \{ \mu_j \}_{\rho(j) = 0} \cup \{ \text{CPA.Enc}(w_{\rho(j)}, \mu_j) \}_{\rho(j) \neq 0} \text{ where } (\{ \mu_j \}, \rho) \leftarrow \text{share}(f, \mu^{(\beta)}) \}
\]

\[
O_X(x) := (ct_x = \{ w_i \}_{x_i = 1})
\]

\[
O_E(i, m) := \text{CPA.Enc}_{w_i}(m)
\]
with the restrictions that (i) only one query is made to each of $\mathcal{O}_F(\cdot)$ and $\mathcal{O}_X(\cdot)$, and (ii) the queries $f$ and $x$ to $\mathcal{O}_F(\cdot)$, $\mathcal{O}_X(\cdot)$ respectively, satisfy $f(x) = 0$.

To be clear, the $\beta$ in $\mathbb{G}^{1\text{-ABE}}_{\beta}$ affects only the implementation of the oracle $\mathcal{O}_F$ (where $\mu^{(\beta)}$ is shared). We will show that $\mathbb{G}^{1\text{-ABE}}_{0} \approx_c \mathbb{G}^{1\text{-ABE}}_1$ where we instantiate $\text{share}$ using the scheme in Section 4.4. That is, Theorem 18 will bound the quantity:

$$\Pr[(A, \mathbb{G}^{1\text{-ABE}}_{0}) = 1] - \Pr[(A, \mathbb{G}^{1\text{-ABE}}_{1}) = 1]$$

**Comparison with [Jaf+17] Adaptive Security of Yao**

**Secret-Sharing**

Proving adaptive security for the core 1-ABE with $\text{share}$ is very similar to the proof for adaptively secure Yao Secret-Sharing for circuits in [Jaf+17]. One main difference is that in our case, the adaptive choices $z$ correspond to both $(f, x)$, while in the adaptive secret-sharing proof of [Jaf+17], $f$ is fixed, and the adaptive choices correspond to $x$, but revealed one bit at a time (that is, $\mathcal{O}_X(i, x_i)$ returns $w_i$ if $x_i = 1$). Another difference is the $\mathcal{O}_E$ oracle included in our core 1-ABE game, which enables the component to be embedded in a standard dual-system hybrid proof for our full ABE systems. Lastly, we leverage our improved analysis in Lemmas 12 and 13 to achieve polynomial security loss, rather than the quasi-polynomial loss we would get from following their proof more directly.

One thing to note is that of these changes, the only one that weakens the result relative to [Jaf+17] is that the adaptive choices defining $x$ are considered in one shot.
(rather than one bit at a time). However, one can consider an alternative version of our core scheme where $O_X(x) := (c'_x = \{w_i\}_{x_i=1})$ is replaced with a suite of oracles $O_X^{(i)} := (w_i)$ with the restriction that, when $x$ is defined to be all 0s except for the bits corresponding to the oracles $O_X^{(i)}$ that were called are set to 1, $f(x) = 0$ still holds. Our game using this oracle suite then allows for adaptive choices of $x$ and is then strictly stronger than adaptive security for Yao Secret Sharing (due to our other change, which requires that security holds even in the presence of the $O_E$ oracle). It is easy to see in the following proof that this oracle suite can just as easily be accommodated in the argument (in Lemma 16, we note that all $w_i$ for which $x_i = 1$ are available, and it is equally easy to give them out individually (as in this proposed change) as it is to give them out as a set (as in the our original definition of $O_X$)). Therefore, this modified argument gives a proof for adaptive security for Yao Secret Sharing for NC$^1$ circuits with the same security loss: one that is polynomial in the security parameter, improving upon the subexponential security loss achieved in [Jaf+17], which posed finding such an improvement as an open question.

**Adaptive Security for Core 1-ABE Component**

We will show that $G_{0-\text{ABE}} \approx_\epsilon G_{1-\text{ABE}}$ as defined in Definition 14 using the Piecewise Guessing framework. To do this, we need to first define a family of games $\{H^u\}$ along with functions $h_0, \ldots, h_L$, using the pebbling strategy in Section 4.3. First, we will describe $\text{share}^u$, which will be used to define $H^u$. 

70
Defining $\text{share}^u$

Recall that Lemma 13 describes how to parse a $u \in \{0, 1\}^{3d}$ as a pebbling configuration: a subset of the nodes of $f$. Further, note that each node contains one output wire, so we can equivalently view $u$ as a subset of $[m]$ denoting the output wires of pebbled gates. Given a pebbling configuration $u$ of an $\text{NC}^1$ formula, the shares are generated as in the secret-sharing scheme in Figure 4.3, except for each pebbled node with output wire $c$, we replace $\mu_c$ with an independent random $\mu_c \leftarrow \mathbb{Z}_p$ (in the case of a pebbled OR gate, we replace both associated $\mu_{ca}$ and $\mu_{cb}$ with independent random $\mu_{ca}, \mu_{cb} \leftarrow \mathbb{Z}_p$, i.e: both $\mu_{ca}, \mu_{cb}$ are associated with wire $c$). In particular, we get the procedure $\text{share}^u(f, \mu)$ defined in Figure 4.5.

$\text{share}^u(f, \mu):$

Input: A formula $f : \{0, 1\}^n \rightarrow \{0, 1\}$, a secret $\mu \in \mathbb{Z}_p$, and a pebbling configuration $u$ of the nodes of $f$.

1. Compute $([\mu'_j], \rho) \leftarrow \text{share}(f, \mu)$ as defined in Figure 4.3
2. For each $\mu'_j$, if $j \in u$ (i.e: if $j$ is the output wire of a pebbled node), then sample $\mu_j \leftarrow \mathbb{Z}_p$. Otherwise, set $\mu_j := \mu'_j$.
3. Output $\{\mu_j\}, \rho.$

Figure 4.5: Pebbling-modified secret sharing scheme $\text{share}^u$

Hybrid Distribution $H^u$

We now define our hybrid games, and remark that Section 4.2 used $z \in \{0, 1\}^R$ to denote the adaptive choices made by an adversary, and the functions $h_t$ that define our hybrid games will depend on the adaptive choices of both the $f \in \text{NC}^1$ and $x \in \{0, 1\}^n$ chosen during the game, so in our application of the Piecewise Guessing
framework of Section 4.2,  \( z \) will be \((f, x)\). Note that the conclusion of the framework is independent of the size of the adaptive input \((R = |f| + n)\), and the framework allows its \( x \) to be defined in parts over time, though in our application, \( x \) will be defined in one shot.

**Definition 15 (\( H^u \) and \( h_\ell \)).** Let \( H^u \) be \( G_0^{1-\text{ABE}} \) with \( \text{share}^u(f, \mu^{(0)}) \) used in the implementation of oracle \( \mathcal{O}_F(f) \) (replacing \( \text{share}(f, \mu^{(0)}) \)). Let \( h_\ell : \mathbf{NC}^1 \times \{0,1\}^n \rightarrow \{0,1\}^R \) denote the function that on formula \( f \) with root gate \( g^* \) and input \( x \in \{0,1\}^n \) where \( f(x) = 0 \), outputs the pebbling configuration created from following the first \( \ell \) instructions from \( \text{Pebble}(f, x, g^*) \) of Figure 4.1.

Note that the first 0 instructions specify a configuration with no pebbles, and all sequences of instructions from \( \text{Pebble}(f, x, g^*) \) when \( f(x) = 0 \) result in a configuration with a single pebble on the root gate \( g^* \), so \( h_0 \) and \( h_L \) are constant functions for all \( f, x \) where \( f(x) = 0 \). Furthermore, note that for all such \( f, x \):

- \( H^{h_0}(f,x) \) is equivalent to \( G_0^{1-\text{ABE}} \) (since \( \text{share}^{h_0(f,x)}(f, \mu^{(0)}) = \text{share}(f, \mu^{(0)}) \));
- \( H^{h_L}(f,x) \) is equivalent to \( G_1^{1-\text{ABE}} \) (since \( \text{share}^{h_L(f,x)}(f, \mu^{(0)}) = \text{share}(f, \mu^{(1)}) \) for an independently random \( \mu^{(1)} \) which is implicitly defined by the independently random value associated with the output wire of the pebbled root gate: \( \mu_m \)).

We now have a series of hybrids \( G_0^{1-\text{ABE}} \equiv H^{h_0}(f,x), H^{h_1}(f,x), \ldots, H^{h_L}(f,x) \equiv G_1^{1-\text{ABE}} \) which satisfy end-point equivalence and, according to the Piecewise Guessing framework described in Section 4.2, define games \( \widehat{H}_{\ell,0}(u_0, u_1), \widehat{H}_{\ell,1}(u_0, u_1) \) for \( \ell \in [0, L] \).
Lemma 16 (neighboring indistinguishability). For all \( \ell \in [L] \) and \( u_0, u_1 \in \{0,1\}^R \),

\[
\Pr[\langle A, \hat{H}_{\ell,0}(u_0, u_1) \rangle = 1] - \Pr[\langle A, \hat{H}_{\ell,1}(u_0, u_1) \rangle = 1] \leq n \cdot \text{Adv}_\text{CPA}^\text{CS} (\lambda)
\]

Proof. First, observe that the difference between \( \hat{H}_{\ell,0}(u_0, u_1) \) and \( \hat{H}_{\ell,1}(u_0, u_1) \) lies in \( \mathcal{O}_F(\cdot) \): the former uses \( \lbrack \text{share}^{u_0} \rbrack \) and the latter uses \( \lbrack \text{share}^{u_1} \rbrack \). Now, fix the adaptive query \( f \) to \( \mathcal{O}_F \). We consider two cases.

First, suppose there does not exist \( x' \in \{0,1\}^n \) such that \( h_{\ell-1}(f, x') = u_0 \) and \( h_{\ell}(f, x') = u_1 \). Then, both \( \langle A, \hat{H}_{\ell,0}(u_0, u_1) \rangle \) and \( \langle A, \hat{H}_{\ell,1}(u_0, u_1) \rangle \) output 0 (i.e., abort) with probability 1 and then we are done.

In the rest of the proof, we deal with the second case, namely there exists \( x' \in \{0,1\}^n \) such that \( h_{\ell-1}(f, x') = u_0 \) and \( h_{\ell}(f, x') = u_1 \). This means that \( u_0 \) and \( u_1 \) are neighboring pebbling configurations in \( \text{Pebble}(f, x', g^*) \), so they differ by a pebbling instruction that follows one of the rules in Definition 9. We proceed via a case analysis depending on what the instruction taking configuration \( u_0 \) to \( u_1 \) is (the instruction is uniquely determined given \( u_0, u_1, f \)):

- pebble/unpebble input node with out-going wire \( \lbrack j \rbrack \). Here, the only difference from \( \text{share}^{u_0}(f, \mu^{(0)}) \) to \( \text{share}^{u_1}(f, \mu^{(0)}) \) is that we change \( \lbrack \mu_j \rbrack \) to a random element of \( \mathbb{Z}_p \) (or vice-versa). The pebbling rule for an input node requires that the input \( x \) to \( \mathcal{O}_X(\cdot) \) in both \( \hat{H}_{\ell,0}(u_0, u_1) \) and \( \hat{H}_{\ell,1}(u_0, u_1) \) satisfies \( x_{\rho(j)} = 0 \). Indistinguishability then follows from the CPA security of (\text{CPA.Setup}, \text{CPA.Enc}, \text{CPA.Dec}) under key \( w_{\rho(j)} \): this is because \( x_{\rho(j)} = 0 \) and therefore \( w_{\rho(j)} \) will not need to be supplied in the answer to the query to \( \mathcal{O}_X(x) \).
In fact, the two hybrids are computationally indistinguishable even if the adversary sees all \( \{w_i : i \neq \rho(j)\} \) (as may be provided by \( \mathcal{O}_X(x) \)).

- pebble/unpebble AND gate with out-going wire \( \overline{e} \) and input wires \( a, b \) corresponding to nodes \( g_a, g_b \). Here, the only difference from \( \text{share}^{u_0}(f, \mu^{(0)}) \) to \( \text{share}^{u_1}(f, \mu^{(0)}) \) is that we change \( \mu_c \) from an actual share \( \hat{\mu}_a + \hat{\mu}_b + \hat{\mu}_c \) to a random element of \( \mathbb{Z}_p \) (or vice-versa). The pebbling rules for an AND gate require that there is a pebble on either \( g_a \) or \( g_b \), say \( g_a \). Therefore, \( \mu_a \) is independent and uniformly random in both distributions \( \text{share}^{u_0}(f, \mu^{(0)}) \) and \( \text{share}^{u_1}(f, \mu^{(0)}) \), and thus \( \hat{\mu}_a \) is fresh and independently random in both distributions (this uses the fact that \( g_a \) has fan-out 1) and makes the distribution of \( \mu_c = \hat{\mu}_a + \hat{\mu}_b + \hat{\mu}_c \) in hybrid \( \ell - 1 \) independently random. We may then deduce that \( \text{share}^{u_0}(f, \mu^{(0)}) \) and \( \text{share}^{u_1}(f, \mu^{(0)}) \) are identically distributed, and therefore so is the output \( \mathcal{O}_F(f) \). (This holds even if the adversary receives all of \( \{w_i : i \in [n]\} \) from its query to \( \mathcal{O}_X(x) \)).

- pebble/unpebble OR gate with out-going wire \( \overline{e} \) and input wires \( a, b \) corresponding to nodes \( g_a, g_b \). Here, the only difference from \( \text{share}^{u_0}(f, \mu^{(0)}) \) to \( \text{share}^{u_1}(f, \mu^{(0)}) \) is that we change \( \mu_{c_a}, \mu_{c_b} \) from actual shares \( (\hat{\mu}_a + \hat{\mu}_c, \hat{\mu}_b + \hat{\mu}_c) \) to random elements of \( \mathbb{Z}_p \) (or vice-versa). The pebbling rules for an OR gate require that there are pebbles on both \( g_a \) and \( g_b \). Therefore, \( \mu_a \) and \( \mu_b \) are independent and uniformly random in both distributions \( \text{share}^{u_0}(f, \mu^{(0)}) \) and \( \text{share}^{u_1}(f, \mu^{(0)}) \), and thus \( \hat{\mu}_a, \hat{\mu}_b \) are fresh and independently random in both distributions (using the fact that \( g_a, g_b \) have fan-out 1), and make the distributions of \( \mu_{c_a} = \hat{\mu}_a + \hat{\mu}_c \), \( \mu_{c_b} = \hat{\mu}_a + \hat{\mu}_b \) in hybrid \( \ell - 1 \) both independently
random. We may then deduce that \( \text{share}^{u_0}(f, \mu^{(0)}) \) and \( \text{share}^{u_1}(f, \mu^{(0)}) \) are identically distributed, and therefore so is the output \( \mathcal{O}_E(f) \). (This holds even if the adversary receives all of \( \{w_i : i \in [n]\} \) in its query to \( \mathcal{O}_X(x) \)).

In all cases, the simulator can return an appropriately distributed answer to \( \mathcal{O}_X(x) = \{w_i\}_{x_i = 1} \) since it has all \( w_i \) except in the first case, where it is missing only a \( w_i \) such that \( x_i = 0 \). Additionally, we note that in all cases, a simulator can return appropriately distributed answers to queries to the encryption oracle \( \mathcal{O}_E(i, m) = \text{Enc}_{w_i}(m) \), since only in the first case (an input node being pebbled or unpebbled) is there a \( w_i \) not directly available to be used to simulate the oracle, and in that case, the simulator has oracle access to an \( \text{Enc}_{w_i}(\cdot) \) function in the CPA symmetric-key security game, and it can uniformly guess which of the \( n \) variables is associated with the input node being pebbled and answer \( \mathcal{O}_E \) requests to that variable with the CPA \( \text{Enc}_{w_i}(\cdot) \) oracle (the factor of \( n \) due to guessing is introduced here since the simulator may not know which variable is associated with the input node at the time of the oracle request, e.g: for requests to \( \mathcal{O}_E \) made before \( \mathcal{O}_X \), so the simulator must guess uniformly and take a security loss of \( n \)).

In all but the input node case, the two distributions \( \langle \mathcal{A}, \hat{H}_{\ell,0}(u_0, u_1) \rangle \) and \( \langle \mathcal{A}, \hat{H}_{\ell,1}(u_0, u_1) \rangle \) are identical, and in the input node case, we've bounded the difference by the distinguishing probability of the symmetric key encryption scheme, the advantage function \( \text{Adv}_{\mathcal{G}}^{\text{CPA}}(\lambda) \), conditioned on a correct guess of which of the \( n \) input variables corre-
sponds to the pebbled/unpebbled input node. Therefore,

\[
\Pr[\langle A, \tilde{H}_{\ell,0}(u_0, u_1) \rangle = 1] - \Pr[\langle A, \tilde{H}_{\ell,1}(u_0, u_1) \rangle = 1] \leq n \cdot \text{Adv}^{\text{CPA}}(\lambda)
\]

\[\square\]

**CPA-secure symmetric encryption**

We will instantiate \(\text{(CPA.Setup, CPA.Enc, CPA.Dec)}\) in our Core 1-ABE of Definition 14 with a variant of the standard CPA-secure symmetric encryption scheme based on \(k\)-Lin from [Esc+13] that supports messages \([M]_2 \in G_2\) of an asymmetric prime-order bilinear group \(G\):

**CPA.Setup\((1^\lambda)\) :** Run \(G \leftarrow G(1^\lambda)\). Sample \(M_0 \leftarrow \mathbb{Z}_p^{k \times k}, m_1 \leftarrow \mathbb{Z}_p^k\).

output \(sk = (sk_0, sk_1) := (M_0, m_1^\top)\)

**CPA.Enc\((sk, [M]_2)\) :** Sample \(r \leftarrow \mathbb{Z}_p^k\), output \((ct_0, ct_1) := ([M + m_1^\top r]_2, [M_0 r]_2)\)

**CPA.Dec\((sk_0, sk_1, (ct_0, ct_1))\) :** Output \(ct_0 \cdot sk_1 \cdot sk_0^{-1} \cdot ct_1\).

**Correctness** Note that: \(ct_0 \cdot sk_1 \cdot sk_0^{-1} \cdot ct_1 = [M + m_1^\top r - m_1^\top r]_2 = [M]_2\).

**Lemma 17.** \(\text{Adv}^{\text{CPA}}(\lambda) \leq \text{Adv}^{k\text{-LIN}}(\lambda)\)

**Proof.** Consider the following adversary \(B^*\), which when given MDDH\(_k\) challenge: \((G, [M]_2, [\tilde{z}]_2)\) first parses \(M\) into \(M_0 := \text{the first } k \text{ rows of } M\) and \(m_1^\top := \text{the } k + 1\text{th row of } M\) (here \([\tilde{z}]_2 = \left(\begin{bmatrix} M_0 & \tilde{s} \\ m_1^\top \end{bmatrix}_2 \right)\) where either \(r \leftarrow \mathbb{Z}_p\) or \(r = 0\) and prepares the following challenge ciphertext using \([\tilde{z}]_2\) from the MDDH\(_k\) challenge: \((ct_0, ct_1) := ([M_0 + m_1^\top \tilde{s} + r]_2, [M_0 \tilde{s}]_2)\).
To answer CPA encryption oracle queries for messages $[M]_2$, draw $r \leftarrow \mathbb{Z}_p^k$ and return $(ct_0, ct_1) = ([M + m_1^r r]_2, [M_0 r]_2)$. Note that this is an appropriately distributed encryption and can be computed using the MDDH$_k$ challenge element $[M]_2$.

If $r = 0$, then the challenge ciphertext is distributed as a normal encryption of $[M]_2$ with $r = s$. This is the normal CPA security game.

If $r \leftarrow \mathbb{Z}_p$, then the challenge ciphertext is distributed as an encryption of a random message $[M + r]_2$ with $r = s$. In this game the adversary’s advantage is 0.

Therefore, we have: $\text{Adv}^{\text{CPA}}_B(\lambda) \leq \text{Adv}^{\text{MDDH}_k}_B(\lambda)$. From Section 2.4 we have that:

$\text{Adv}^{\text{MDDH}_k}_B(\lambda) = \text{Adv}^{\text{K-LIN}}_B(\lambda)$ So, we have: $\text{Adv}^{\text{CPA}}_B(\lambda) \leq \text{Adv}^{\text{K-LIN}}_B(\lambda)$. ☐

**Theorem 18.** The Core 1-ABE component of Definition 14 implemented with (share, reconstruct) from Section 4.4 and the CPA-secure symmetric encryption scheme (CPA.Setup, CPA.Enc, CPA.Dec) from Section 4.4 satisfies:

$$\Pr[\langle A, G_0^{1-\text{ABE}} \rangle = 1] - \Pr[\langle A, G_1^{1-\text{ABE}} \rangle = 1] \leq 2^{6d} \cdot 8^d \cdot n \cdot \text{Adv}^{\text{K-LIN}}_B(\lambda)$$

**Proof.** Recall the hybrids $G_0^{1-\text{ABE}} \equiv H_{h_0(f,x)}, H_{h_1(f,x)}, ..., H_{h_L(f,x)} \equiv G_1^{1-\text{ABE}}$ defined in Section 4.4. Lemma 16 tells us that: for all $\ell \in [L]$ and $u_0, u_1 \in \{0, 1\}^{R'}$,

$$\Pr[\langle A, \hat{H}_{\ell,0}(u_0, u_1) \rangle = 1] - \Pr[\langle A, \hat{H}_{\ell,1}(u_0, u_1) \rangle = 1] \leq n \cdot \text{Adv}^{\text{CPA}}_B(\lambda)$$

These hybrids satisfy the end-point equivalence requirement, so Lemma 8 then
tells us that:

$$\Pr[\langle A, G_0^{1-\text{ABE}} \rangle = 1] - \Pr[\langle A, G_1^{1-\text{ABE}} \rangle = 1] \leq 2^{2R'} \cdot L \cdot n \cdot \text{Adv}_B^{\text{CPA}}(\lambda)$$

Lemma 13 tells us that $R' \leq 3d$, and Lemma 11 tells us that $L \leq 8^d$, where $d$ is the depth of the formula. Finally, Lemma 17 tells us that $\text{Adv}_B^{\text{CPA}}(\lambda) \leq \text{Adv}_B^{k-\text{LIN}}(\lambda)$. So:

$$\Pr[\langle A, G_0^{1-\text{ABE}} \rangle = 1] - \Pr[\langle A, G_1^{1-\text{ABE}} \rangle = 1] \leq 2^{6d} \cdot 8^d \cdot n \cdot \text{Adv}_B^{k-\text{LIN}}(\lambda)$$

\[\square\]

### 4.5 KP-ABE for Boolean Formulas

In this section, we present our compact KP-ABE for Boolean Formulas that is adaptively secure under the MDDH$_k$ assumption in asymmetric prime-order bilinear groups. For attributes of length $n$, our ciphertext comprises $O(n)$ group elements, independent of the formula size, while simultaneously allowing attribute reuse in the formula. As mentioned in the overview in Section 4.1, we incorporated optimizations from [Gon+16; BKP14] to shrink $W_i$ and thus the secret key, and hence the need for the $O_E$ oracle in the core 1-ABE security game.

**The scheme**

Our KP-ABE scheme is as follows:
Setup(1^λ, 1^n) : Run \( \mathcal{G} = (p, G_1, G_2, G_T, e) \leftarrow \mathcal{G}(1^\lambda) \). Sample

\[
\begin{align*}
A &\leftarrow \mathbb{Z}_p^{k \times (k+1)}, \quad W_i \leftarrow \mathbb{Z}_p^{(k+1) \times k} \quad \forall i \in [n], \quad v \leftarrow \mathbb{Z}_p^{k+1}
\end{align*}
\]

and output:

\[
\begin{align*}
\text{msk} &:= (v, W_1, \ldots, W_n) \\
\text{mpk} &:= ([A]_1, [AW]_1, \ldots, [AW]_n, e([A]_1, [v]_2)
\end{align*}
\]

Enc(mpk, x, M) : Sample \( s \leftarrow \mathbb{Z}_p^k \). Output:

\[
\begin{align*}
\text{ct}_x &:= (ct_1, \{ct_{2,i}\}_{x_i=1}, ct_3) \\
&:= \begin{pmatrix} [s^\top A]_1, \{[s^\top AW]_i\}_{x_i=1}, & e([s^\top A]_1, [v]_2) \cdot M \end{pmatrix}
\end{align*}
\]

KeyGen(mpk, msk, f) : Sample \( \{v_j\}, \rho \leftarrow \text{share}(f, v) \), \( r_j \leftarrow \mathbb{Z}_p^k \). Output:

\[
\begin{align*}
\text{sk}_f &:= \{sk_{1,j}, sk_{2,j}\} \\
&:= \{[v_j + W_{\rho(j)} r_j]_2, [r_j]_2\}
\end{align*}
\]

where \( W_0 = 0 \).

Dec(mpk, sk_f, ct_x) : Compute \( \omega_j \) such that \( v = \sum_{\rho(j)=0 \land \rho(j)=1} \omega_j v_j \) as described in
Section 4.4. Output:

\[
ct_3 \cdot \prod_{\rho(j) = 0 \lor x_{\rho(j)} = 1} \left( \frac{e(ct_{2,\rho(j)}, sk_{2,j})}{e(ct_1, sk_{1,j})} \right)^{\omega_j}
\]

Correctness

Correctness relies on the fact that for all \( j \), we have

\[
\frac{e(ct_1, sk_{1,j})}{e(ct_{2,\rho(j)}, sk_{2,j})} = [s^T A v_j]_T
\]

which follows from the fact that

\[
s^T A v_j = s^T_A \cdot (v_j + W_{\rho(j)} r_j) - s^T Aw_{\rho(j)} \cdot r_j
\]

Therefore, for all \( f, x \) such that \( f(x) = 1 \), we have:

\[
ct_3 \cdot \prod_{\rho(j) = 0 \lor x_{\rho(j)} = 1} \left( \frac{e(ct_{2,\rho(j)}, sk_{2,j})}{e(ct_1, sk_{1,j})} \right)^{\omega_j} = M \cdot [s^T A v_j]_T \cdot \prod_{\rho(j) = 0 \lor x_{\rho(j)} = 1} [s^T A v_j]_T^{\omega_j}
\]

\[
= M \cdot [s^T A v_j]_T \cdot [-s^T A \sum_{\rho(j) = 0 \lor x_{\rho(j)} = 1} \omega_j v_j]_T
\]

\[
= M \cdot [s^T A v_j]_T \cdot [-s^T A v]_T
\]

\[
= M
\]

80
Adaptive Security

Description of hybrids

To describe the hybrid distributions, it would be helpful to first give names to the various forms of ciphertext and keys that will be used. A ciphertext can be in one of the following forms:

- **Normal**: generated as in the scheme.

- **SF**: same as a Normal ciphertext, except $s^\top A$ replaced with $c^\top \leftarrow \mathbb{Z}_p^{k+1}$. That is,

$$ct_x := \left( [c_1^\top, \{ [c_j^\top W_i] \}_{i=1}^1, e([c_1^\top, [v]_2 : M] \right)$$

A secret key can be in one of the following forms:

- **Normal**: generated as in the scheme.

- **SF**: same as a Normal key, except $v$ replaced with $v + \delta a^\perp$, where a fresh $\delta \leftarrow \mathbb{Z}_p$ is chosen per SF key and $a^\perp$ is any fixed $a^\perp \in \mathbb{Z}_p^{k+1} \setminus \{0\}$ such that $Aa^\perp = 0$.

That is,

$$sk_f := ( \{ [v_j + W_{\rho(j)} r_j]_2, [r_j]_2 \} \right)$$

where $(\{v_j\}, \rho) \leftarrow \text{share}(f, [v + \delta a^\perp]), r_j \leftarrow \mathbb{Z}_p^k$.

SF stands for semi-functional following the terminology in previous works [LW10; Wat09].
Hybrid sequence.

Suppose the adversary A makes at most $Q$ secret key queries. The hybrid sequence is as follows:

- $H_0$: real game
- $H_1$: same as $H_0$, except we use a SF ciphertext.
- $H_{2,\ell}, \ell = 0, \ldots, Q$: same as $H_1$, except the first $\ell$ keys are SF and the remaining $Q - \ell$ keys are Normal.
- $H_3$: replace $M$ with random $\tilde{M}$.

Proof overview.

- We have $H_0 \approx_c H_1 \equiv H_{2,0}$ via $k$-Lin, which tells us $([A]_1, [s^T A]_1) \approx_c ([A]_1, [c^\top]_1)$. Here, the security reduction will pick $W_1, \ldots, W_n$ and $v$ so that it can simulate the $\text{mpk}$, the ciphertext and the secret keys.
- We have $H_{2,\ell-1} \approx_c H_{2,\ell}$, for all $\ell \in [Q]$. The difference between the two is that we switch the $\ell$th $sk_f$ from Normal to SF using the adaptive security of our core 1-ABE component in $G^{1-ABE}$ from Section 4.4.

The idea is to sample

$$v = \tilde{v} + \mu a^\perp, W_i = \tilde{W}_i + a^\perp w_i^\top$$

so that $\text{mpk}$ can be computed using $\tilde{v}, \tilde{W}_i$ and perfectly hide $\mu, w_1, \ldots, w_n$. Roughly speaking: the reduction
- uses $O_X(x)$ in $G^{1-ABE}$ to simulate the challenge ciphertext
- uses $O_F(f)$ in $G^{1-ABE}$ to simulate $\ell$th secret key
- uses $\mu^{(0)}$ from $G^{1-ABE}$ together with $O_E(i, \cdot) = Enc(w_i, \cdot)$ to simulate the remaining $Q - \ell$ secret keys

- We have $H_{2,Q} \equiv H_3$. In $H_{2,Q}$, the secret keys only leak $v + \delta_1 a^\perp, \ldots, v + \delta_Q a^\perp$.

This means that $c^T v$ is statistically random (as long as $c^T a^\perp \neq 0$).

**Lemma 19 ($H_0 \approx_c H_1 \equiv H_{2,0}$).**

$$|Pr[\langle A, H_0 \rangle = 1] - Pr[\langle A, H_1 \rangle = 1]| \leq \text{Adv}^{k-\text{LIN}}_{A'} (\lambda)$$

**Proof.** Given MDDH$_k$ challenge ($[A]_1, [z^T]_1$), where either $z^T = s^T A$ or $z^T = c^T$, an adversary $A'$ could simply choose $W_i \leftarrow Z_p^{(k+1)\times k}$, $v \leftarrow Z_p^{k+1}$, form the public parameters with $A, W_i, v$, and choose its own $r_j \leftarrow Z_p^k$ when responding to key requests. For the challenge ciphertext, $A'$ creates:

$$ct_x := \left( [z^T]_1, \{[z^T W_i]_1\}_{x_i = 1}, e([z^T]_1, [v]_2) \cdot M \right)$$

If $z^T = s^T A$, $A'$ has simulated $H_0$; If $z^T = c^T$, $A'$ has simulated $H_1 \equiv H_{2,0}$.

**Lemma 20 ($H_{2,\ell-1} \approx_c H_{2,\ell}$).**

$$|Pr[\langle A, H_{2,\ell-1} \rangle = 1] - Pr[\langle A, H_{2,\ell} \rangle = 1]| \leq 2^{6d} \cdot 8^d \cdot n \cdot \text{Adv}^{k-\text{LIN}}_{B^*} (\lambda)$$

**Proof.** For each $\beta \in \{0, 1\}$, consider the following adversary $A'$ in $G^{1-ABE}_\beta$ which inter-
nally simulates \( \mathcal{A} \) and the challenger in the ABE security game:

- First, \( \mathcal{A}' \) samples \( \mathbf{A} \leftarrow \mathbb{Z}_p^{k\times(k+1)} \), \( \mathbf{W}_i \leftarrow \mathbb{Z}_p^{(k+1)\times k} \), \( \tilde{\mathbf{v}} \leftarrow \mathbb{Z}_p^{k+1} \), computes \( \mathbf{a}^\perp \in \mathbb{Z}_p^{k+1} \setminus \{0\} \) such that \( \mathbf{Aa}^\perp = \mathbf{0} \) and implicitly defines

\[
\mathbf{v} := \tilde{\mathbf{v}} + \mathbf{\mu}^{(0)} \mathbf{a}^\perp, \quad \mathbf{W}_i := \tilde{\mathbf{W}}_i + \mathbf{a}^\perp \mathbf{w}_i^\top
\]

where \( \mathbf{w}_i \in \mathbb{Z}_p^k, \mathbf{\mu}^{(0)} \in \mathbb{Z}_p \) are chosen in \( \mathcal{G}_{\beta}^{1-\text{ABE}} \). Then, \( \mathcal{A}' \) outputs:

\[
\text{mpk} := ( [\mathbf{A}]_1, [\mathbf{A}\tilde{\mathbf{W}}_1]_1, \ldots, [\mathbf{A}\tilde{\mathbf{W}}_n]_1, e([\mathbf{A}]_1, [\tilde{\mathbf{v}}]_2) )
\]

- When \( \mathcal{A} \) requests a challenge ciphertext for attribute \( \mathbf{x} \) along with \( M_0, M_1 \), \( \mathcal{A}' \) queries \( \mathcal{O}_X(\mathbf{x}) \rightarrow ( \{ \mathbf{w}_i \}_{x_i=1} ) \) in \( \mathcal{G}_{\beta}^{1-\text{ABE}} \). \( \mathcal{A}' \) then samples \( \mathbf{c} \leftarrow \mathbb{Z}_p^{k+1} \) and \( b \leftarrow \{0, 1\} \) (the challenge bit in the standard ABE security game) and returns the following (SF) challenge ciphertext for \( \mathcal{A} \):

\[
\text{ct}_x = \left( [\mathbf{c}]_1, \{ [\mathbf{c}^\top (\tilde{\mathbf{W}}_i + \mathbf{a}^\perp \mathbf{w}_i^\top)]_{x_i=1} \}, e([\mathbf{c}]_1, [\mathbf{\tilde{v}} + \mathbf{\mu}^{(0)} \mathbf{a}^\perp]_2) \cdot M_b \right)
\]

- For the first \( \ell - 1 \) secret keys requested, say for formula \( f \), \( \mathcal{A}' \) computes

\[
([\mathbf{v}_j], \rho) \leftarrow \text{share}(f, \tilde{\mathbf{v}} + \tilde{\delta} \mathbf{a}^\perp)
\]

where \( \tilde{\delta} \leftarrow \mathbb{Z}_p \) is drawn independently for each key (here, the per-key \( \delta = \tilde{\delta} - \mathbf{\mu}^{(0)} \) implicitly). Next, for each \( j \), it queries \( \mathcal{O}_E(\rho(j), [0]_2) \rightarrow ([\mathbf{w}^\top_{\rho(j)} \mathbf{r}_j]_2, [\mathbf{r}_j]_2) \) in \( \mathcal{G}_{\beta}^{1-\text{ABE}} \) (since \( \mathcal{O}_E(\rho(j), [0]_2) = \text{CPA.Enc}_{\mathbf{w}_{\rho(j)}}([0]_2) \)), and forms the following (SF)
key:

\[ \text{sk}_f = ( \{ [v_j + \tilde{W}_{\rho(j)} r_j + a^\perp \tilde{w}_{\rho(j)}^\top r_j], [r_j]_2 \} ) \]

- For the last \( Q - \ell \) secret keys requested, say for formula \( f \), \( \mathcal{A}' \) proceeds as before for the first \( \ell - 1 \) keys except

\[ (\{ v_j \}, \rho) \leftarrow \text{share}(f, \tilde{v} + \mu^{(0)} a^\perp) \]

It is easy to see that it forms a Normal key.

- For the \( \ell \)th secret key requested, say for formula \( f \), \( \mathcal{A}' \) computes \((\{ v_j \}, \rho) \leftarrow \text{share}(f, \tilde{v})\), queries \( O_f(f) \rightarrow (\{ [\mu_j + w_{\rho(j)}^\top r_j], [r_j]_2 \}) \) in \( G_\beta^{1-\text{ABE}} \), then uses these components to return:

\[ \text{sk}_f = ( \{ [v_j + \tilde{W}_{\rho(j)} r_j + a^\perp (\mu_j + w_{\rho(j)}^\top r_j)], [r_j]_2 \} ) \]

\[ = (v_j + \mu_j a^\perp) + w_j r_j \]

We claim that if \( \beta = 0 \), then \( \text{sk}_f \) is a Normal key, and if \( \beta = 1 \), then \( \text{sk}_f \) is a SF key. This follows the fact that thanks to linearity, the shares

\[ (\{ v_j + \mu_j a^\perp \}, \rho), \text{ where } (\{ v_j \}, \rho) \leftarrow \text{share}(f, \tilde{v}), (\{ v_j \}, \rho) \leftarrow \text{share}(f, \mu^{(\beta)}) \]

are identically distributed to \( \text{share}(f, \tilde{v} + \mu^{(\beta)} a^\perp) \). The claim then follows from the fact that \( \tilde{v} + \mu^{(0)} a^\perp = v \) and that \( \tilde{v} + \mu^{(1)} a^\perp \) is identically distributed to \( v + \delta a^\perp \) (where \( \delta = \mu^{(1)} - \mu^{(0)} \) is a fresh random value for this key).
Putting everything together, for $\beta \in \{0, 1\}$, when $\mathcal{A}'$ interacts with $G_{\beta}^{1\text{-ABE}}$, then $\mathcal{A}'$ simulates $H_{2,\ell-1+\beta}$. It follows then that:

$$|\Pr[\langle \mathcal{A}, H_{2,\ell-1} \rangle = 1] - \Pr[\langle \mathcal{A}, H_{2,\ell} \rangle = 1]| \leq |\Pr[\langle \mathcal{A}', G_{\beta}^{1\text{-ABE}} \rangle = 1] - \Pr[\langle \mathcal{A}', G_{1}^{1\text{-ABE}} \rangle = 1]|$$

From Theorem 18, we then have:

$$|\Pr[\langle \mathcal{A}, H_{2,\ell-1} \rangle = 1] - \Pr[\langle \mathcal{A}, H_{2,\ell} \rangle = 1]| \leq 2^{6d} \cdot 8^d \cdot n \cdot \text{Adv}_{G_{\beta}}^{k\text{-LIN}} (\lambda)$$

\[\square\]

**Lemma 21** ($H_{2,Q} \approx_s H_3$).

$$|\Pr[\langle \mathcal{A}, H_{2,Q} \rangle = 1] - \Pr[\langle \mathcal{A}, H_3 \rangle = 1]| \leq \frac{1}{p}$$

**Proof.** These two hybrids are identically distributed conditioned on $c^\top a^\perp \neq 0$. To see this, consider two ways of sampling $v$: as $\tilde{v} \leftarrow \mathbb{Z}_p^{k+1}$ and as $\tilde{v} + \tilde{m}a^\perp$ for an independent $\tilde{m} \leftarrow \mathbb{Z}_p$. Note that both result in $v$ having a uniform distribution.

Using $\tilde{v}$ to simulate hybrid $H_{2,Q}$ obviously results in $H_{2,Q}$ (where $v = \tilde{v}$). However, using the identically distributed $v = \tilde{v} + \tilde{m}a^\perp$ to simulate $H_{2,Q}$ results in $H_3$ (where $\tilde{M} = M \cdot e([c^\top], [\tilde{m}a^\perp])$ is randomly distributed as long as $c^\top a^\perp \neq 0$, and for redefined independently random $\tilde{\delta}_i := \delta_i + \tilde{m}$ in the secret keys).

$c$ is chosen at random and independent from $a^\perp \neq 0$, so $c^\top a^\perp = 0$ with probability
and since we know that $H_{2,Q} \equiv H_3$ conditioned on $c^\top a^\perp \neq 0$, then we have:

$$|\Pr[\langle A, H_{2,Q} \rangle = 1] - \Pr[\langle A, H_3 \rangle = 1]| \leq \frac{1}{p}$$

\[ \square \]

**Theorem 22** (adaptive KP-ABE). *The KP-ABE construction in Section 4.5 is adaptively secure under the MDDH$_k$ assumption.*

**Proof.**

$$|\Pr[\langle A, H_0 \rangle = 1] - \Pr[\langle A, H_3 \rangle = 1]| \leq |\Pr[\langle A, H_0 \rangle = 1] - \Pr[\langle A, H_1 \rangle = 1]|$$

$$+ \sum_{\ell=1}^{Q} |\Pr[\langle A, H_{2,\ell-1} \rangle = 1] - \Pr[\langle A, H_{2,\ell} \rangle = 1]|$$

$$+ |\Pr[\langle A, H_{2,Q} \rangle = 1] - \Pr[\langle A, H_3 \rangle = 1]|$$

(Since $H_1 \equiv H_{2,0}$). Summing the results of Lemmas 19, 20, and 21, we then have:

$$|\Pr[\langle A, H_0 \rangle = 1] - \Pr[\langle A, H_3 \rangle = 1]| \leq \text{Adv}_{G^*}^{k-LIN}(\lambda) + Q \cdot 2^{6d} \cdot 8^d \cdot n \cdot \text{Adv}_{G^*}^{k-LIN}(\lambda) + \frac{1}{p}$$

If $d = O(\log n)$, then under the $k$-Lin Assumption this is a negligible function of $\lambda$ (the number of queries made $Q$ and the attribute vector length $n$ are both polynomial in $\lambda$, and $\frac{1}{p}$ is a negligible function of $\lambda$). It’s easy to see that $\text{Adv}_A^{\text{ABE}}(\lambda) = 0$ in the $H_3$ hybrid game (since a random message is encrypted in the challenge ciphertext). So, any adversary in the real game ($H_0$) will have advantage negligibly close to 0, and
our construction satisfies adaptive security.

4.6 CP-ABE for Boolean Formulas

In this section, we present a compact CP-ABE for Boolean formulas that is adaptively secure under the MDDH$_k$ assumption in asymmetric prime-order bilinear groups. The construction is analogous to our KP-ABE scheme in Section 4.5. The reduction is also a bit more complex as we need to embed the output of $O_T$ into the CP-ABE ciphertext. This is facilitated by an additional dimension of isolation provided by $Br$ in the secret keys, which requires additional intermediate distributions in the proof of security (and also removes the use of the $O_E$ oracle in the core 1-ABE security game).

CP-ABE Construction

Our CP-ABE scheme is as follows:

Setup($1^\lambda, 1^n$) : Run $G = (p, G_1, G_2, G_T, e) \leftarrow G(1^\lambda)$. Sample

$$A \leftarrow \mathbb{Z}_p^{k \times 2k}, B \leftarrow \mathbb{Z}_p^{(k+1) \times k}, U_0, W_i \leftarrow \mathbb{Z}_p^{2k \times (k+1)}, v \leftarrow \mathbb{Z}_p^{2k}$$
and output:

\[
\begin{align*}
\text{msk} & := (v, B, U_0, W_1, \ldots, W_n) \\
\text{mpk} & := ([A]_1, [AU_0]_1, [AW_1]_1, \ldots, [AW_n]_1, e([A]_1, [v]_2))
\end{align*}
\]

Enc(mpk, f, M) : Sample \((\{u_j^\top\}, \rho) \leftarrow \text{share}(f, s^\top AU_0), s, s_j \leftarrow \mathbb{Z}_p^k\). Output:

\[
\begin{align*}
\text{ct}_f & = (\text{ct}_1, \{\text{ct}_{2,j}, \text{ct}_{3,j}\}, \text{ct}_4) \\
& := \left([s^\top A]_1, \{[u_j^\top + s_j^\top AW_{\rho(j)}]_1, [s_j^\top A]_1\}, e([s^\top A]_1, [v]_2) \cdot M\right)
\end{align*}
\]

where \(W_0 = 0\).

KeyGen(mpk, msk, f) : Sample \(r \leftarrow \mathbb{Z}_p^k\). Output:

\[
\begin{align*}
\text{sk}_x & = (\text{sk}_1, \text{sk}_2, \{\text{sk}_{3,i}\}_{x_i=1}) \\
& := (v + U_0 Br]_2, [Br]_2, \{[W_i Br]_2\}_{x_i=1})
\end{align*}
\]

Dec(mpk, skx, ct_f) : Compute \(\omega_j\) such that \(s^\top AU_0 = \sum_{\rho(j) = 0 \lor x_{\rho(j)} = 1} \omega_j u_j\) as described in Section 4.4. Output:

\[
\frac{\text{ct}_4}{e(\text{ct}_1, \text{sk}_1)} \cdot \prod_{\rho(j) = 0 \lor x_{\rho(j)} = 1} \left( \frac{e(\text{ct}_{2,j}, \text{sk}_2)}{e(\text{ct}_{3,j}, \text{sk}_{3,\rho(j)})} \right)^{\omega_j}
\]
Correctness

Correctness relies on the fact that for all $j$, we have

$$\frac{e(ct_{2,j}, sk_2)}{e(ct_{3,j}, sk_{3,\rho(j)})} = [u_j^\top Br]_T$$

which follows from the fact that

$$u_j^\top Br = (u_j^\top + s_j^\top AW_{\rho(j)}) \cdot \underbrace{Br - s_j^\top A \cdot W_{\rho(j)} Br}_{ct_{2,j} \text{ sk}_2}$$

and also from the fact that

$$e(ct_1, sk_1) = [s^\top Av + s^\top AU_0 Br]_T$$

Therefore, for all $f, x$ such that $f(x) = 1$, we have:

$$\frac{ct_1}{e(ct_1, sk_1)} \cdot \prod_{\rho(j)=0 \lor x_{\rho(j)}=1} \left( \frac{e(ct_{2,j}, sk_2)}{e(ct_{3,j}, sk_{3,\rho(j)})} \right)^{\omega_j} = \frac{M \cdot [s^\top Av]_T}{[s^\top Av + s^\top AU_0 Br]_T} \cdot \prod_{\rho(j)=0 \lor x_{\rho(j)}=1} [u_j^\top Br]_T^{\omega_j}$$

$$= \frac{M \cdot [s^\top Av]_T}{[s^\top Av + s^\top AU_0 Br]_T} \cdot \sum_{\rho(j)=0 \lor x_{\rho(j)}=1} \omega_j u_j^\top Br_T$$

$$= \frac{M \cdot [s^\top Av + s^\top AU_0 Br]_T}{[s^\top Av + s^\top AU_0 Br]_T}$$

$$= M$$

90
Adaptive Security

Description of hybrids

A ciphertext can be in one of the following forms:

- **Normal**: generated as in the scheme.

- **SF**: same as a Normal ciphertext, except \( s^\top A, s_j^\top A \) replaced with \( c^\top, c_j^\top \), where \( c, c_j \leftarrow \mathbb{Z}_p^{2k} \). That is,

\[
ct_f := \left( \left[ c^\top \right]_1, \{ [u_j^\top + c_j^\top W_{\rho(j)}]_1, [c_j^\top]_1 \}, e([c^\top]_1, [v]_2 \cdot M) \right)
\]

A secret key can be in one of the following forms:

- **Normal**: generated as in the scheme.

- **SF**: same as a Normal key, except \( v \) replaced with \( v + A^\perp \delta(q) \), where a fresh \( \delta(q) \leftarrow \mathbb{Z}_p^k \) is chosen per SF and \( A^\perp \) is any fixed \( A^\perp \in \mathbb{Z}_p^{2k \times k} \setminus \{0\} \) such that \( AA^\perp = 0 \). That is,

\[
\text{sk}_x := \left( \left[ v + A^\perp \delta(q) \right] + U_0 Br \right)_2, \left[ Br \right]_2, \{ \left[ W_i Br \right]_2 \}_{x_i=1} \)
\]

- **P-Normal**: same as a Normal key, except \( Br \) replaced with \( d \leftarrow \mathbb{Z}_p^{k+1} \). That is,

\[
\text{sk}_x := \left( \left[ v + U_0 d \right]_2, \left[ d \right]_2, \{ \left[ W_i d \right]_2 \}_{x_i=1} \right)
\]
• P-SF: same as a SF key, except Br replaced with \( d \leftarrow \mathbb{Z}_p^{k+1} \). That is,

\[
\text{sk}_x := (v + A \cdot \delta^{(q)} + U_0[d]_2, [d]_2, \{[W_i d]_2\}_{x_i=1})
\]

Here, P stands for pseudo following [Wee14; CGW15].

**Hybrid sequence.**

Suppose the adversary A makes at most \( Q \) secret key queries. The hybrid sequence is as follows:

• \( H_0 \): real game

• \( H_1 \): same as \( H_0 \), except we use a SF ciphertext.

• \( H_{2,\ell,1}, \ell = 0, \ldots, Q \): same as \( H_1 \), except the \( \ell \)'th key is P-Normal, the first \( \ell - 1 \) keys are SF and the last \( Q - \ell \) keys are Normal.

• \( H_{2,\ell,2} \): same as \( H_{2,\ell,1} \) except the \( \ell \)'th key is P-SF.

• \( H_{2,\ell,3} \): same as \( H_{2,\ell,1} \) except the \( \ell \)'th key is SF.

• \( H_3 \): replace \( M \) with random.

**Proof overview.**

• We have \( H_0 \approx_c H_1 \equiv H_{2,0,3} \) via \( k\)-Lin (and its self-reducibility), which tells us

\[
([A]_1, [s^T A]_1, \{[s^T_j A]_1\}) \approx_c ([A]_1, [c^T]_1, \{[c^T_j]_1\})
\]
Here, the security reduction will pick $\mathbf{U}_0, \mathbf{W}_1, \ldots, \mathbf{W}_n$ and $\mathbf{v}$ so that it can simulate the mpk, the ciphertext and the secret keys.

- We have $H_{2,\ell-1,3} \approx_c H_{2,\ell,1}$ for all $\ell \in [Q]$. The difference between the two is that we switch the $\ell$th $\mathbf{sk}_f$ from Normal to P-Normal.

This follows again via $k$-Lin, which tells us $([\mathbf{B}]_2, [\mathbf{Br}]_2) \approx_c ([\mathbf{B}]_2, [\mathbf{d}]_2)$. Again, the security reduction will pick $\mathbf{U}_0, \mathbf{W}_1, \ldots, \mathbf{W}_n$ and $\mathbf{v}$ so that it can simulate the mpk, the ciphertext and the secret keys.

- We have $H_{2,\ell,1} \approx_c H_{2,\ell,2}$ for all $\ell \in [Q]$. The difference between the two is that we switch the $\ell$th $\mathbf{sk}_f$ from P-Normal to P-SF. The idea is to program:

$$W_i = \tilde{W}_i + \mathbf{A}^\perp \mathbf{w}_i (\mathbf{b}^\perp)^\top, \quad U_0 = \tilde{U}_0 + \mathbf{A}^\perp \mathbf{u} (\mathbf{b}^\perp)^\top$$

where $\mathbf{w}_i, \mathbf{b}^\perp \in \mathbb{Z}_p^k, \mathbf{A}^\perp \in \mathbb{Z}_p^{2k \times k}, \mathbf{u} \in \mathbb{Z}_p^{k+1}$ and

$$\mathbf{A} \mathbf{A}^\perp = 0, (\mathbf{b}^\perp)^\top \mathbf{B} = 0$$

Note that the public parameters and the normal and SF keys information-theoretically hide a random $\mathbf{u}$ and the $\mathbf{w}_i$’s from $G^{1-\text{ABE}}$. Next, we focus on the SF ciphertext $\mathbf{ct}_f$, and the $\ell$th secret key $\mathbf{sk}_x$, which is either P-Normal or P-SF.

First, we argue that if we ignore $\mathbf{v} + \mathbf{U}_0 \mathbf{d}$ in $\mathbf{sk}_x$, then $\mathbf{u}$ remains computationally hidden given $\mathbf{ct}_f, \mathbf{sk}_x$ using the $G^{1-\text{ABE}}$ security game. Theorem 18 tells us that
\( \mathbf{u} \) is computationally hidden given

\[
c_c \{ [c_j^T \mathbf{A}^\perp]_{i=1}, [\mu_j + c_j^T \mathbf{A}^\perp \mathbf{w}_{\rho(j)}]_{j}, \{ \mathbf{w}_i \}_{x_i=1} \}
\]

where \( \{ \{ \mu_j \}, \rho \} \leftarrow \text{share}(f, c^T \mathbf{A}^\perp \mathbf{u}) \) and we treat \( c_j^T \mathbf{A}^\perp \in \mathbb{Z}_p^{1 \times k}, c_j \leftarrow \mathbb{Z}_p^{2k} \) as the randomness used for CPA.Enc, even for adaptive choices of \( f, x \). We can then use the entropy in \( \mathbf{u} \) to hide the \( \mathbf{A}^\perp \)-component of \( \mathbf{v} \) in \( \mathbf{v} + \mathbf{A}^\perp \mathbf{u} (b^\perp)^T \mathbf{d} \).

- We have \( H_{2, \ell, 2} \approx_c H_{2, \ell, 3} \) for all \( \ell \in [Q] \). The difference between the two is that we switch the \( \ell \)th \( \text{sk}_j \) from \( \text{P-SF} \) to \( \text{SF} \).

This follows again via \( k \)-Lin, which tells us \( ([\mathbf{B}]_2, [\mathbf{B}r]_2) \approx_c ([\mathbf{B}]_2, [\mathbf{d}]_2) \), symmetrically to the proof for \( H_{2, \ell-1, 3} \approx_c H_{2, \ell, 1} \), except that \( \mathbf{v} + \mathbf{A}^\perp \mathbf{d}^{(\ell)} \) is used instead of \( \mathbf{v} \) in the \( \ell \)th secret key.

- We have \( H_{2, Q, 3} \equiv_c H_3 \). In \( H_{2, Q, 3} \), the secret keys only leak \( \mathbf{v} + \mathbf{A}^\perp \mathbf{d}^{(1)}, \ldots, \mathbf{v} + \mathbf{A}^\perp \mathbf{d}^{(Q)} \). This means that \( c^T \mathbf{v} \) is statistically random (as long as \( c^T \mathbf{A}^\perp \neq \mathbf{0} \)).

**Lemma 23** (\( H_0 \approx_c H_1 \equiv H_{2, 0, 3} \)).

\[
| \Pr[\langle \mathcal{A}, H_0 \rangle = 1] - \Pr[\langle \mathcal{A}, H_1 \rangle = 1] | \leq \text{Adv}_{A^*}^{k-\text{LIN}}(\lambda)
\]

**Proof.** Given MDDH\(_{2m+1}^{k, 2k} \) challenge \( ([\mathbf{A}]_1, [\mathbf{Z}]_1)_1 \), where either \( \mathbf{Z}^T = \mathbf{S}^T \mathbf{A} \) for a \( \mathbf{S}^T \leftarrow \mathbb{Z}_p^{(2m+1) \times k} \) or \( \mathbf{Z}^T = \mathbf{C}^T \) for a \( \mathbf{C}^T \leftarrow \mathbb{Z}_p^{(2m+1) \times 2k} \), an adversary \( \mathcal{A}' \) could simply choose \( \mathbf{U}_0, \mathbf{W}_i \leftarrow \mathbb{Z}_p^{2k \times (k+1)}, \mathbf{v} \leftarrow \mathbb{Z}_p^{2k} \), form the public parameters with \( \mathbf{A}, \mathbf{U}_0, \mathbf{W}_i, \mathbf{v} \), choose its own \( \mathbf{B} \leftarrow \mathbb{Z}_p^{(k+1) \times k}, \mathbf{r} \leftarrow \mathbb{Z}_p^k \) when responding to key requests.

For the challenge ciphertext, \( \mathcal{A}' \) computes: \( \{ [\mathbf{u}_j^T] \}, \rho \) \leftarrow \text{share}(f, z_2^{\mathbf{U}_0}, \mathbf{U}_0), \) parses
the rows of $Z^T$ as $z_j^T$ for $j \in [m+1]$, and returns:

$$ct_γ := \left([z_{2m+1}]_1, [u_j^T + z_j^T W_{\rho(j)}]_1, [z_j^T]_1, e([z_{2m+1}]_1; [v]_2) \cdot M_b\right)$$

(note that $|\{u_j\}| \leq 2m$).

If $Z^T = S^T A$, then the challenge ciphertext is Normal and $A'$ has simulated $H_0$;

If $Z^T = C^T$, then the challenge ciphertext is SF and $A'$ has simulated $H_1 \equiv H_{2,0,3}$.

Finally, recall from Section 2.4 that $\text{Adv}_{A'}^{\text{MDDH}^{2m+1}_{k,2^k}} (\lambda) = \text{Adv}_{A^*}^{k\text{-LIN}} (\lambda)$

\[\square\]

**Lemma 24** ($H_{2, \ell-1, 3} \approx_c H_{2, \ell, 1}$).

$$|\text{Pr}[\langle A, H_{2, \ell-1, 3} \rangle = 1] - \text{Pr}[\langle A, H_{2, \ell, 1} \rangle = 1]| \leq \text{Adv}_{A^*}^{k\text{-LIN}} (\lambda)$$

**Proof.** Given MDDH$_k$ challenge $([B]_2, [z]_2)$, where either $z = Br$ for $r \leftarrow Z_p^k$ or $z = d$, for $d \leftarrow Z_p^{k+1}$, an adversary $A'$ could simply choose $A \leftarrow Z_p^{k \times 2^k}$, $U_0, W_i \leftarrow Z_p^{2k \times (k+1)}$, $v \leftarrow Z_p^{2k}$, and form the public parameters with $A, U_0, W_i, v$. $A'$ could then compute $A^\perp \in Z_p^{2k \times k} \setminus \{0\}$ such that $AA^\perp = 0$ (to be used in answering secret key queries).

- For the (SF) challenge ciphertext, $A'$ computes: $([u_j^T], \rho) \leftarrow \text{share}(f, c^T U_0)$, draws $c, c_j \leftarrow Z_p^{2k}$ for each $j$, and creates:

$$ct_f := \left([c^T]_1, [u_j^T + c_j^T W_{\rho(j)}]_1, [c_j^T]_1, e([c^T]_1; [v]_2) \cdot M_b\right)$$
• For the first $\ell - 1$ secret keys requested, say the $q$th request is for $x$, $A'$ draws $\delta^{(q)}, r^{(q)} \leftarrow \mathbb{Z}_p^k$, and forms the following (SF) key:

$$sk_x := ( [v + A^{\perp} \delta^{(q)} + U_0 Br^{(q)}]_2, [Br^{(q)}]_2, \{[W_i Br^{(q)}]_2\} x_i = 1 )$$

• For the last $Q - \ell$ secret keys requested, $A'$ proceeds as before for the first $\ell - 1$ keys except using just $v$ instead of $v + A^{\perp} \delta^{(q)}$. It is easy to see that it forms a Normal key.

• For the $\ell$th secret key request, $A'$ forms the following key:

$$sk_x := ( [v + U_0 z]_2, [z]_2, \{[W_i z]_2\} x_i = 1 )$$

If $z = Br$, then the $\ell$th key is Normal and $A'$ has simulated $H_{2,\ell-1,3}$.

If $z = d$, then the $\ell$th key is $P$-Normal and $A'$ has simulated $H_{2,\ell,1}$.

\[\square\]

**Lemma 25** ($H_{2,\ell,1} \approx_c H_{2,\ell,2}$).

$$| \Pr[\langle A,H_{2,\ell,1}\rangle = 1] - \Pr[\langle A,H_{2,\ell,2}\rangle = 1]| \leq 2^{6d} \cdot 8^d \cdot n \cdot \text{Adv}_{kLIN}^1(\lambda)$$

**Proof.** Consider the following adversary $A'$ in $C^1_{beta}$ which internally simulates $A$ and the challenger in the ABE security game:

• First, $A'$ samples $A \leftarrow \mathbb{Z}_p^{k \times 2k}, B \leftarrow \mathbb{Z}_p^{(k+1) \times k}, \tilde{U}_0, \tilde{W}_i \leftarrow \mathbb{Z}_p^{2k \times (k+1)}, \tilde{v} \leftarrow \mathbb{Z}_p^{2k}$, computes $A^{\perp} \setminus \{0\} \in \mathbb{Z}_p^{2k \times k}, b^{\perp} \in \mathbb{Z}_p^{(k+1)}$ such that $AA^{\perp} = 0$ and $(b^{\perp})^T B = 0$
and implicitly defines
\[ v := \bar{v} - \frac{\mu^{(0)}((b^\perp)^T d)}{(c^T A^\perp u)} A^\perp u, U_0 := \bar{U}_0 + \frac{\mu^{(b)}}{(c^T A^\perp u)} A^\perp u (b^\perp)^T, W_i := \bar{W}_i + A^\perp w_i (b^\perp)^T \]

where \( w_i \in \mathbb{Z}_p^k, \mu^{(b)} \in \mathbb{Z}_p \) are chosen in \( G_\beta^{1-\text{ABE}} \), \( c \leftarrow \mathbb{Z}_p^{2k} \) is chosen for use in the challenge ciphertext, \( d \leftarrow \mathbb{Z}_p^{k+1} \) is chosen for use in the \( \ell \text{th} \) secret key, and \( u \leftarrow \mathbb{Z}_p^k \). Note that \( A' \) can compute \( v \) since it has \( \mu^{(0)} \) from \( G_\beta^{1-\text{ABE}} \) and knows all other vectors.

Then, \( A' \) generates the public parameters as:

\[ \text{mpk} := ( [A]_1, [A \bar{U}_0]_1, [A \bar{W}_1]_1, \ldots, [A \bar{W}_n]_1, e([A]_1, [\bar{v}]_2) ) \]

**When \( A \) requests a challenge ciphertext for formula \( f \) along with \( M_0, M_1, A' \) queries \( O_f(f) \rightarrow ([\mu_j + r_j^T w_{\rho(j)}], [r_j]_1) \) in \( G_\beta^{1-\text{ABE}} \). \( A' \) then samples \( \tilde{c}_j \leftarrow \mathbb{Z}_p^k \) for each \( j \) and \( b \leftarrow \{0, 1\} \) (the challenge bit in the standard ABE security game), defines \( A^\perp_C := \begin{bmatrix} (A^\perp)^T \\ M \end{bmatrix} \in \mathbb{Z}_p^{2k \times 2k} \) for a choice of \( M \) that makes \( A^\perp_C \) invertible, computes \( [c_j]_1 := (A^\perp_C)^{-1} \begin{bmatrix} r_j \\ \tilde{c}_j \end{bmatrix} \), computes: \( ([u_j]^T, \rho) \leftarrow \text{share}(f, c^T \bar{U}_0) \), and returns the following appropriately distributed (SF) challenge ciphertext:

\[ \text{ct}_f := \begin{cases} [c^T]_1, ([u_j]^T + (\mu_j + r_j^T w_{\rho(j)})(b^\perp)^T + c_j^T \bar{W}_{\rho(j)}]_1, [c_j]_1, e([c^T]_1, [v]_2) \cdot M_b \end{cases} \]

\[ = \begin{cases} [c^T]_1, ([u_j]^T + c_j^T \bar{W}_{\rho(j)} + r_j^T w_{\rho(j)}(b^\perp)^T]_1, [c_j]_1, e([c^T]_1, [v]_2) \cdot M_b \end{cases} \]
Note that \( \{ \mu_j(b^\perp)^T \} \) is distributed like the output of \( \text{share}(f, \mu(b)(b^\perp)^T) \), and therefore due to linearity and the fact that \( c^T U_0 = c^T \tilde{U}_0 + \mu(b)(b^\perp)^T \), then
\[
\{ \tilde{u}_j^T + \mu_j(b^\perp)^T \} \text{ is distributed like } \text{share}(f, c^T \tilde{U}_0 + \mu(b)(b^\perp)^T) = \text{share}(f, c^T U_0).
\]
Also, note that \( c_j^T W_i = c_j^T \tilde{W}_i + r_j w_\rho(j) b^\perp \) since \( c_j^T A^\perp = r_j \).

- For the first \( \ell - 1 \) secret keys requested, say the \( q \)th request is for \( x, A' \) draws \( \tilde{\delta}^{(q)}, r^{(q)} \leftarrow \mathbb{Z}_p^k \), and forms the following (SF) key:

\[
\text{sk}_x = ( \begin{bmatrix} v + A^\perp \tilde{\delta}^{(q)} + \tilde{U}_0 Br^{(q)} \end{bmatrix}_2, [Br^{(q)}]_2, \{ \tilde{W}_i Br^{(q)} \}_{x_i=1} )
\]

- For the last \( Q - \ell \) secret keys requested, \( A' \) proceeds as before for the first \( \ell - 1 \) keys except using just \( v \) instead of \( v + A^\perp \tilde{\delta}^{(q)} \). It is easy to see that it forms a Normal key.

- For the \( \ell \)th secret key requested, say for \( x \), queries \( \mathcal{O}_x(x) \rightarrow ( \{ w_i \}_{x_i=1} ) \) in \( G_{\beta}^{1-\text{ABE}} \), then uses these components to return:

\[
\text{sk}_x = ( \begin{bmatrix} \tilde{v} + \tilde{U}_0 d \end{bmatrix}_2, [d]_2, \{ \left[ \tilde{W}_i + A^\perp w_i (b^\perp)^T d \right]_{x_i=1} \})
\]

If \( \beta = 0 \), then the \( \ell \)th key is a P-Normal key since
\[
v + \frac{(\mu(0) - \mu(1))((b^\perp)^T d) A^\perp u}{(u^\perp A^\perp u)} A^\perp u + U_0 d = v.
\]

If \( \beta = 1 \), then the \( \ell \)th key is a P-SF key, where \( \tilde{\delta}^{(\ell)} = \frac{(\mu(0) - \mu(1))((b^\perp)^T d)}{(u^\perp A^\perp u)} d. \)

Putting everything together, for \( \beta \in \{0, 1\} \), when \( A' \) interacts with \( G_{\beta}^{1-\text{ABE}} \), then \( A' \) simulates \( H_{2,\ell,1+\beta} \). So:

\[
|\Pr[\langle A, H_{2,\ell,1} \rangle = 1] - \Pr[\langle A, H_{2,\ell,2} \rangle = 1]| \leq |\Pr[\langle A', G_{0}^{1-\text{ABE}} \rangle = 1] - \Pr[\langle A', G_{1}^{1-\text{ABE}} \rangle = 1]|.
\]

98
and from Theorem 18, we then have:

\[ | \Pr[\langle \mathcal{A}, H_{2,\ell-1} \rangle = 1] - \Pr[\langle \mathcal{A}, H_{2,\ell} \rangle = 1] | \leq 2^{6d} \cdot 8^d \cdot n \cdot \text{Adv}_{G^*}^{k-LIN}(\lambda) \]

\[ \square \]

**Lemma 26** \((H_{2,\ell,2} \approx_c H_{2,\ell,3})\).

\[ | \Pr[\langle \mathcal{A}, H_{2,\ell,2} \rangle = 1] - \Pr[\langle \mathcal{A}, H_{2,\ell,3} \rangle = 1] | \leq \text{Adv}_{\mathcal{A}}^{k-LIN}(\lambda) \]

**Proof.** Omitted, since the proof is completely analogous to that of Lemma 24, using \(v + A^\perp \tilde{\delta}(\ell)\) for a new random \(\tilde{\delta}(\ell)\) instead of \(v\) when creating the \(\ell\)th key. \(\square\)

**Lemma 27** \((H_{2,Q,3} \approx_c H_3)\).

\[ | \Pr[\langle \mathcal{A}, H_{2,Q,3} \rangle = 1] - \Pr[\langle \mathcal{A}, H_3 \rangle = 1] | \leq \frac{1}{p} \]

**Proof.** These two hybrids are identically distributed conditioned on \(c^TA^\perp \neq \tilde{0}\). To see this, consider two ways of choosing \(v\): \(v = \tilde{v} \leftarrow \mathbb{Z}_p^{2k}\) and \(v = \tilde{v} + A^\perp \tilde{m}\) for an independently random \(\tilde{m} \leftarrow \mathbb{Z}_p^k\). Note that both result in \(v\) having a uniform distribution.

Using \(\tilde{v}\) to simulate hybrid \(H_{2,Q,3}\) obviously results in \(H_{2,Q,3}\) (where \(v = \tilde{v}\)). However, using the identically distributed \(v = \tilde{v} + A^\perp \tilde{m}\) to simulate \(H_{2,Q,3}\) results in \(H_3\) (where \(M \cdot [c^T A^\perp \tilde{m}]_T\) is a randomly distributed message as long as \(c^T A^\perp \neq \tilde{0}\), and for redefined independently random \(\tilde{\delta}^{(i)} = \tilde{\delta}^{(i)} + \tilde{m}\) in the secret keys).
\( c \) is chosen at random and independent from \( A^\perp \neq 0 \), so \( c^\top A^\perp = 0 \) with probability \( \frac{1}{p} \), and since we know that \( H_{2,Q,3} \equiv H_3 \) conditioned on \( c^\top A^\perp = 0 \), then we have:

\[
|\Pr[\langle A, H_{2,Q,3} \rangle = 1] - \Pr[\langle A, H_3 \rangle = 1]| \leq \frac{1}{p}
\]

\[\,\square\,\]

**Theorem 28** (adaptive CP-ABE). The CP-ABE construction in Appendix 4.6 is adaptively secure under the MDDH\(_k\) assumption.

**Proof.** Note that since \( H_1 \equiv H_{2,0,3} \):

\[
|\Pr[\langle A, H_0 \rangle = 1] - \Pr[\langle A, H_3 \rangle = 1]| \leq |\Pr[\langle A, H_0 \rangle = 1] - \Pr[\langle A, H_1 \rangle = 1]|
\]

\[
+ \sum_{\ell=1}^Q |\Pr[\langle A, H_{2,\ell-1,3} \rangle = 1] - \Pr[\langle A, H_{2,\ell,1} \rangle = 1]|\]

\[
+ \sum_{\ell=1}^Q |\Pr[\langle A, H_{2,\ell,1} \rangle = 1] - \Pr[\langle A, H_{2,\ell,2} \rangle = 1]|\]

\[
+ \sum_{\ell=1}^Q |\Pr[\langle A, H_{2,\ell,2} \rangle = 1] - \Pr[\langle A, H_{2,\ell,3} \rangle = 1]|\]

\[
+ |\Pr[\langle A, H_{2,Q,3} \rangle = 1] - \Pr[\langle A, H_3 \rangle = 1]|\]

Summing the results of Lemmas 23, 24, 25, 26, and 27, we then have that:

\[
|\Pr[\langle A, H_0 \rangle = 1] - \Pr[\langle A, H_3 \rangle = 1]| \leq Adv_{\overline{G}^*}^{k-LIN} (\lambda) + 2 \cdot Q \cdot Adv_{\overline{G}^*}^{k-LIN} (\lambda) + Q \cdot 2^{6d} \cdot 8^d \cdot n \cdot Adv_{\overline{G}^*}^{k-LIN} (\lambda) + \frac{1}{p}
\]

If \( d = O(\log n) \), then under the \( k\)-Lin assumption this quantity is a negligible function of \( \lambda \) (the number of queries made \( Q \) and the attribute vector length \( n \) are
both polynomial in $\lambda$, and $\frac{1}{p}$ is a negligible function of $\lambda$). It’s easy to see that $\text{Adv}_{A}^{\text{ARE}}(\lambda) = 0$ in the $H_3$ hybrid game (since a random message is encrypted in the challenge ciphertext). So, any adversary in the real game ($H_0$) will have advantage negligibly close to 0, and our construction satisfies adaptive security. \qed
Chapter 5

*Putting It All Together*

5.1 Unbounded KP-ABE for Boolean formulas

**Overview**

In this section, we use the modular technique from [Che+18] presented in Section 3.3 to transform our KP-ABE construction from Section 4.5 (for NC^1 that is compact and adaptively secure under the MDDH_k assumption in asymmetric prime-order bilinear groups) into a construction with the same properties plus an added benefit that the scheme is unbounded (i.e: the public parameters are of constant size). Our ease in doing so demonstrates the benefit of our modular approach to unboundedness. We note that a similar application can transform our CP-ABE from Section 4.6 into an unbounded variant.

**Unbounded KP-ABE Construction**

\[ \text{Setup}(1^\lambda, 1^n) : \text{Run } G = (p, G_1, G_2, G_T, e) \leftarrow G(1^\lambda). \text{ Sample} \]

\[ A_1 \leftarrow \mathbb{Z}_p^{k \times (2k+1)}, W, W_0, W_1 \leftarrow \mathbb{Z}_p^{(2k+1) \times k}, v \leftarrow \mathbb{Z}_p^{2k+1} \]
and output:

\[
\text{msk} := (\mathbf{v}, \mathbf{W}, \mathbf{W}_0, \mathbf{W}_1)
\]

\[
\text{mpk} := ([A_1]_1, [A_1 \mathbf{W}]_1, [A_1 \mathbf{W}_0]_1, [A_1 \mathbf{W}_1]_1, e([A_1]_1, [\mathbf{v}]_2)
\]

\[
\text{Enc}(\text{mpk}, x, M) : \text{Sample } s, s_i \leftarrow \mathbb{Z}_p^k. \text{ Output:}
\]

\[
\text{ct}_x = (\text{ct}_1, \{\text{ct}_{2,i}, \text{ct}_{3,i}\}_{x_i=1}, \text{ct}_4)
\]

\[
:= ([s^\top A_1]_1, \{[s^\top A_1 \mathbf{W} + s_i^\top A_1(\mathbf{W}_0 + i \cdot \mathbf{W}_1)]_1, [s_i^\top A_1]_1\}_{x_i=1}, e([s^\top A_1]_1, [\mathbf{v}]_2 \cdot M)
\]

\[
\text{KeyGen}(\text{mpk}, \text{msk}, f) : \text{Sample } (\{\mathbf{v}_j\}, \rho) \leftarrow \text{share}(f, \mathbf{v}), r_j \leftarrow \mathbb{Z}_p^k. \text{ Output:}
\]

\[
\text{sk}_f = (\{sk_{1,j}, sk_{2,j}, sk_{3,j}\}, \{sk_{4,j}\})
\]

\[
:= (\{[\mathbf{v}_j + \mathbf{W} r_j]_2, [r_j]_2, ([\mathbf{W}_0 + \rho(j) \cdot \mathbf{W}_1] r_j]_2\}_{\rho(j)\neq 0}, \{[\mathbf{v}_j]_2\}_{\rho(j)=0})
\]

\[
\text{Dec}(\text{mpk}, \text{sk}_f, \text{ct}_x) : \text{Compute } \omega_j \text{ such that } \mathbf{v} = \sum_{\rho(j)=0, x_{\rho(j)}=1}^{} \omega_j \mathbf{v}_j \text{ as described in Section 4.4. Output:}
\]

\[
\text{ct}_4 \cdot \prod_{x_{\rho(j)}=1} e(\text{ct}_{2,\rho(j)}, \text{sk}_{2,j})^{\omega_j} \cdot \prod_{\rho(j)=0} e(\text{ct}_1, \text{sk}_{4,j})^{-\omega_j}
\]
Correctness

Correctness relies on the fact that for all \( j \), we have

\[
\frac{e(\text{ct}_1, \text{sk}_{1,j}) \cdot e(\text{ct}_{3,\rho(j)}, \text{sk}_{3,j})}{e(\text{ct}_{2,\rho(j)}, \text{sk}_{2,j})} = [s^\top A_1 v_j]_T
\]

which follows from the fact that

\[
s^\top A_1 v_j = s^\top A_1 \cdot (v_j + W r_j)
\]

\[
- \left( s^\top A_1 W + s^\top A_1 (W_0 + \rho(j) \cdot W_1) \right) \cdot r_j
\]

\[
+ s^\top A_1 \cdot (W_0 + \rho(j) \cdot W_1) r_j^2
\]

and also the fact that for all \( j \), \( e(\text{ct}_1, \text{sk}_{4,j}) = [s^\top A_1 v_j]_T \).

Therefore, for all \( f, x \) such that \( f(x) = 1 \), we have:

\[
\text{ct}_4 \cdot \prod_{x_{\rho(j)} = 1} \left( \frac{e(\text{ct}_{2,\rho(j)}, \text{sk}_{2,j})}{e(\text{ct}_1, \text{sk}_{1,j}) \cdot e(\text{ct}_{3,\rho(j)}, \text{sk}_{3,j})} \right) ^{\omega_j} \cdot \prod_{\rho(j) = 0} e(\text{ct}_1, \text{sk}_{4,j})^{-\omega_j}
\]

\[
= M \cdot [s^\top A_1 v_j]_T \cdot \prod_{\rho(j) = 0 \vee x_{\rho(j)} = 1} \left[ s^\top A_1 v_j \right]_T^{-\omega_j}
\]

\[
= M \cdot [s^\top A_1 v_j]_T \cdot \prod_{\rho(j) = 0 \vee x_{\rho(j)} = 1} \omega_j v_j T
\]

\[
= M \cdot [s^\top A_1 v_j]_T \cdot [-s^\top A_1] \sum_{\rho(j) = 0 \vee x_{\rho(j)} = 1} \omega_j v_j T
\]

\[
= M \cdot [s^\top A_1 v_j]_T \cdot [-s^\top A_1 v]_T
\]

\[
= M
\]
Adaptive Security

Entropy expansion lemma.

Our security proof relies on the “entropy expansion lemma” in [Che+18], presented in its composite order form in Section 3.3. We will use a translation of the lemma into the prime order setting, using a similar framework as the one described in Section 4.1.

First, we introduce some additional notation. Let $A$ be a matrix over $\mathbb{Z}_p$. We use $\text{span}(A)$ to denote the column span of $A$, and we use $\text{span}^\ell(A)$ to denote matrices of width $\ell$ where each column lies in $\text{span}(A)$; this means $M \leftarrow_r \text{span}^\ell(A)$ is a random matrix of width $\ell$ where each column is chosen uniformly from $\text{span}(A)$. We use $\text{basis}(A)$ to denote a basis of $\text{span}(A)$, and we use $(A_1 \mid A_2)$ to denote the concatenation of matrices $A_1, A_2$.

Pick random

$$A_1 \leftarrow_r \mathbb{Z}_p^{\ell_1 \times \ell}, A_2 \leftarrow_r \mathbb{Z}_p^{\ell_2 \times \ell}, A_3 \leftarrow_r \mathbb{Z}_p^{\ell_3 \times \ell}$$

where $\ell := \ell_1 + \ell_2 + \ell_3$. Let $(A_1^\parallel | A_2^\parallel | A_3^\parallel)^T$ denote the inverse of $(A_1^T | A_2^T | A_3^T)^T$, so that $A_i^\parallel A_i^\parallel = I$ (known as non-degeneracy) and $A_i^\parallel A_j^\parallel = 0$ if $i \neq j$ (known as orthogonality). Here, we focus on the case $(\ell_1, \ell_2, \ell_3) = (k, 1, k)$ and so $\ell = 2k + 1$.

**Lemma 29** (entropy expansion lemma [Che+18]). Under the MDDH$_k$ assumption, we have

$$\mathbb{D}_0 := \left\{ \begin{array}{l} \text{aux} : \quad [A_1]_1, [A_1 W]_1, [A_1 W_0]_1, [A_1 W_1]_1 \\ \text{ct} : \quad [c^T]_1, \quad \{ [c^T W + c_i^T (W_0 + i \cdot W_1)]_1, \quad [c_i^T]_1 \}_{i \in [n]} \\ \text{sk} : \quad \{ [WD_i]_2, [D_i]_2, [(W_0 + i \cdot W_1) D_i]_2 \}_{i \in [n]} \end{array} \right\}$$
\[
\approx_c \\
\begin{cases}
\text{aux} : & [A_1]_1, [A_1 W]_1, [A_1 W_0]_1, [A_1 W_1]_1 \\
\text{ct} : & \left[ c^T \right]_1, \\
& \left\{ \left[ c^T (W + V_i^{(2)}) + c_i^T (W_0 + i \cdot W_1 + U_i^{(2)}) \right]_1, \left[ c_i^T \right]_1 \right\}_{i \in [n]} \\
\text{sk} : & \left\{ \left[ (W + V_i^{(2)}) D_i \right]_2, \left[ D_i \right]_2, \left[ (W_0 + i \cdot W_1 + U_i^{(2)}) D_i \right]_2 \right\}_{i \in [n]} 
\end{cases}
\]

where \( W, W_0, W_1 \leftarrow_r \mathbb{Z}_p^{(2k+1) \times k} \), \( V_i^{(2)}, U_i^{(2)} \leftarrow_r \text{span}^k(A_2^\|), D_i \leftarrow_r \mathbb{Z}_p^{k \times k} \), and \( c, c_i \leftarrow_r \text{span}(A_1^\top) \) in the left distribution while \( c, c_i \leftarrow_r \text{span}(A_1^\top, A_2^\top) \) in the right distribution, where the concrete security loss \( |\text{Pr}[A'(D_0) = 1] - \text{Pr}[A'(D_1) = 1]| \leq (5n + 1) \cdot \text{Adv}^{k-\text{LIN}}_G (\lambda) \).

This lemma allows us to use a hybrid proof to first transition to a game in which the challenge ciphertext and secret keys have components in the \( A_2 \) space which mirror those of our (bounded) construction of Section 4.5. We then follow the same proof structure as in Section 4.5.

Description of hybrids

A ciphertext can be in one of the following forms:

- **Normal**: generated as in the scheme.
- **SF**: same as a **Normal** ciphertext, except \( s_i^T A_1, s_i^T A_1 \) replaced with \( c_i^T, c_i^T \leftarrow \mathbb{Z}_p^{2k+1} \) and we use the substitution:

\[
\begin{align*}
W & \rightarrow \tilde{V}_i := W + V_i^{(2)} \text{ in } i\text{'th component, and} \\
W_0 + i \cdot W_1 & \rightarrow \tilde{U}_i := W_0 + i \cdot W_1 + U_i^{(2)}
\end{align*}
\] (5.1)
where \( \mathbf{U}_i^{(2)}, \mathbf{V}_i^{(2)} \leftarrow \text{span}^k(\mathbf{A}_2^\parallel) \). Concretely, a SF ciphertext is of the form:

\[
\text{ct}_x := ( [c^\top]_1, \{ [c^\top \mathbf{V}_i + \mathbf{c}_i^\top \mathbf{U}_i]_1, [c_i^\top]_1 \}_{x_i=1}, e([c^\top]_1, [\mathbf{v}]_2) : M )
\]

A secret key can be in one of the following forms:

- **Normal**: generated as in the scheme.
- **P-SF**: same as a **Normal** key, except we use the same substitution as in (5.1), concretely making a P-SF key of the form:

\[
\text{sk}_f := ( \{ [\mathbf{v}_j + \mathbf{\hat{V}}_{\rho(j)} \mathbf{r}_j]_2, [\mathbf{r}_j]_2, [\mathbf{\hat{U}}_{\rho(j)} \mathbf{r}_j]_2 \}_{\rho(j)\neq 0}, \{ [\mathbf{v}_j]_2 \}_{\rho(j)=0} )
\]

- **SF**: same as a P-SF key, except \( \mathbf{v} \) replaced with \( \mathbf{v} + \mathbf{a}^\perp \), where a fresh \( \delta \leftarrow \mathbb{Z}_p \) is chosen per SF key and \( \mathbf{a}^\perp \leftarrow \text{span}(\mathbf{A}_2^\parallel) \setminus \{ \mathbf{0} \} \).

**Hybrid sequence.**

Suppose the adversary \( \mathcal{A} \) makes at most \( Q \) secret key queries. The hybrid sequence is as follows:

- **H_0**: real game
- **H_1**: same as \( H_0 \), except all keys are P-SF, and we use a SF ciphertext.
- **H_{2,\ell}, \ell = 0, \ldots, Q**: same as \( H_1 \), except the first \( \ell \) keys are SF and the remaining \( Q - \ell \) keys are P-SF.
- **H_3**: replace \( M \) with random \( \tilde{M} \).
Proof overview.

- We have $H_0 \approx_c H_1 \equiv H_{2,0}$ via Lemma 29. In the reduction, on input

$$
\begin{align*}
\text{aux} & : \ [A_1^\top]_1, [A_1^\top W]_1, [A_1^\top W_0]_1, [A_1^\top W_1]_1 \\
\text{ct} & : \ [C_0]_1, \ \{[C_{1,i}]_1, [C_{2,i}]_1\}_{i \in \{n\}} \\
\text{sk} & : \ \{[K_{0,i}]_2, [K_{1,i}]_2, [K_{2,i}]_2\}_{i \in \{n\}}
\end{align*}
$$

we sample $v \leftarrow \mathbb{Z}_p^{2k+1}$, compute $(\{v_j\}, \rho) \leftarrow \text{share}(f, v)$, draw $r_{j,\ell} \leftarrow \mathbb{Z}_p^k$ for shares $j$ and keys $\ell \in [Q]$, and simulate the game with

$$
\begin{align*}
\text{mpk} & : \ \text{aux}, e([A_1^\top]_1, [v]_2) \\
\text{ct}_x & : \ [C_0]_1, \ \{[C_{1,i}]_1, [C_{2,i}]_1\}_{i \not= 1}, e([C_0]_1, [v]_2) \cdot M_b \\
\text{sk}_f^\ell & : \ \{[v_k + K_{0,\rho(j)} r_{j,\ell}]_2, [K_{1,\rho(j)} r_{j,\ell}]_2, [K_{2,\rho(j)} r_{j,\ell}]_2\}
\end{align*}
$$

In both cases, we set $r_{j,\ell} := D_{\rho(j)} r_{j,\ell}$ where $D_{\ell} \leftarrow \mathbb{Z}_p^{k \times k}$ as defined in the entropy expansion lemma (Lemma 29). Therefore all $r_{j,\ell}$ are uniformly distributed over $\mathbb{Z}_p^k$ with high probability.

- We have $H_{2,\ell-1} \approx_c H_{2,\ell}$, for all $\ell \in [Q]$. The difference between the two is that we switch the $\ell$th $\text{sk}_f$ from $P$-$\text{SF}$ to $\text{SF}$ using the adaptive security of our core 1-ABE component in $G^{1-\text{ABE}}$ from Section 4.4.

The idea is to sample

$$v = \bar{v} + \mu a^\perp$$

where $a^\perp \leftarrow \text{span}(A_2^\parallel) \setminus \{0\}$ so that $\text{mpk}$ can be computed using $\bar{v}$ and perfectly
hides $\mu, w_1, \ldots, w_n$. Roughly speaking: the reduction

- uses $O_X(x)$ in $G^{1-\text{ABE}}$ to simulate the challenge ciphertext
- uses $O_F(f)$ in $G^{1-\text{ABE}}$ to simulate $\ell$th secret key
- uses $\mu^{(0)}$ from $G^{1-\text{ABE}}$ together with $O_E(i, \cdot) = \text{Enc}(w_i, \cdot)$ to simulate the remaining $Q - \ell$ secret keys

- We have $H_{2,Q} \equiv H_3$. In $H_{2,Q}$, the secret keys only leak $v + \delta_1 a^\perp, \ldots, v + \delta_Q a^\perp$.

This means that $c^T v$ is statistically random (as long as $c^T a^\perp \neq 0$).

**Lemma 30** ($H_0 \approx_c H_1 \equiv H_{2,0}$).

$$|\Pr[\langle A, H_0 \rangle = 1] - \Pr[\langle A, H_1 \rangle = 1]| \leq (5n + 1) \cdot \text{Adv}_{BF}^{k-\text{LIN}}(\lambda)$$

**Proof.** Consider the following adversary $A'$ attempting to distinguish the distributions in the Entropy Expansion Lemma 29, which internally simulates $A$ and the challenger in the ABE security game:

- $A'$ receives input:

\[
\mathbb{D}_\beta = \begin{cases}
\text{aux} : & [A_1^T]_1, [A_1^T W]_1, [A_1^T W_0]_1, [A_1^T W_1]_1 \\
\text{ct} : & [C_0]_1, \{[C_{1,i}]_1, [C_{2,i}]_1\}_{i \in [n]} \\
\text{sk} : & \{[K_{0,i}]_2, [K_{1,i}]_2, [K_{2,i}]_2\}_{i \in [n]} \\
\end{cases}
\]

- First, $A'$ samples $v \leftarrow_r \mathbb{Z}_p^{2k+1}$ and outputs:

\[
\text{mpk} := ([A_1]_1, [A_1 W]_1, [A_1 W_0]_1, [A_1 W_1]_1, e([A_1]_1, [v]_2)),
\]

109
• When $A$ requests a challenge ciphertext for attribute $x$ along with $M_0, M_1, A'$ samples $b \leftarrow \{0, 1\}$ (the challenge bit in the standard ABE security game) and returns the following challenge ciphertext for $A$:

$$\text{ct}_x := [C_0]_1, \{[C_{1,i}]_1, [C_{2,i}]_1\}_{i:x_i=1}, e([C_0]_1, [v]_2) \cdot M_b$$

• For any secret keys requested, say for formula $f$, $A'$ computes $(\{v_j\}, \rho) \leftarrow \text{share}(f, v)$, draws $\tilde{r}_j \leftarrow \mathbb{Z}_p^k$ and forms the following key:

$$\text{sk}_f := \{[v_j + K_{0,\rho(j)}\tilde{r}_j]_2, [K_{1,\rho(j)}\tilde{r}_j]_2, [K_{2,\rho(j)}\tilde{r}_j]_2\}$$

Notice that if $\beta = 0$ (the input to $A'$ was drawn from distribution $D_0$ defined in Lemma 29), then the challenge $\text{ct}_x$ and all $\text{sk}_f$ are Normal, and if $\beta = 1$ (the input to $A'$ was drawn from distribution $D_1$), then $\text{ct}_x$ is distributed as a SF ciphertext and all $\text{sk}_f$ are distributed as P-SF keys.

Putting everything together, for $\beta \in \{0, 1\}$, when $A'$ interacts with $D_\beta$, then $A'$ simulates $H_\beta$. It follows then that:

$$|\Pr[\langle A, H_0 \rangle = 1] - \Pr[\langle A, H_1 \rangle = 1]| \leq |\Pr[A'(D_0) = 1] - \Pr[A'(D_1) = 1]|$$

From Lemma 29, we then have:

$$|\Pr[\langle A, H_0 \rangle = 1] - \Pr[\langle A, H_1 \rangle = 1]| \leq (5n + 1) \cdot \text{Adv}_{k}^{\text{Lin}}(\lambda)$$
Lemma 31 \( (H_{2,\ell-1} \approx_c H_{2,\ell}) \).

\[
| \Pr[\langle A, H_{2,\ell-1} \rangle = 1] - \Pr[\langle A, H_{2,\ell} \rangle = 1] | \leq 2^{6d} \cdot 8^d \cdot n \cdot Adv^{k-\text{LIN}}_V (\lambda)
\]

Proof. For each \( \beta \in \{0,1\} \), consider the following adversary \( \mathcal{A}' \) in \( G_{\beta}^{1-\text{ABE}} \) which internally simulates \( \mathcal{A} \) and the challenger in the ABE security game:

- First, \( \mathcal{A}' \) samples \( A_1, A_3 \leftarrow \mathbb{Z}_p^{k\times(2k+1)}, A_2 \leftarrow \mathbb{Z}_p^{1\times(2k+1)}, W, W_0, W_1 \leftarrow \mathbb{Z}_p^{(2k+1)\times k}, \tilde{v} \leftarrow \mathbb{Z}_p^{2k+1} \), samples \( (U_i^{(2)} \in \mathbb{Z}_p^{(2k+1)\times k}), (\tilde{V}_i^{(2)} \in \mathbb{Z}_p^{(2k+1)\times k}) \leftarrow \text{span}^k(A_2^\parallel) \) and \( (a^\perp \in \mathbb{Z}_p^{2k+1}) \leftarrow \text{span}(A_2^\parallel) \setminus \{\bar{0}\} \), and implicitly defines

\[
v := \tilde{v} + \mu^{(0)} a^\perp, \quad V_i^{(2)} := \tilde{V}_i^{(2)} + a^\perp w_i
\]

where \( \mu^{(0)} \in \mathbb{Z}_p, w_i \in \mathbb{Z}_p^k \) is chosen in \( G_{\beta}^{1-\text{ABE}} \). (Note that \( v \) is distributed randomly in \( \mathbb{Z}_p^{2k+1} \) and \( V_i^{(2)} \) is distributed like the output of \( \text{span}^k(A_2^\parallel) \)). Then, \( \mathcal{A}' \) outputs:

\[
\text{mpk} := ( [A_1]_1, [A_1W]_1, [A_1W_0]_1, [A_1W_1]_1, e([A_1]_1, [\tilde{v}]_2) ),
\]

- When \( \mathcal{A} \) requests a challenge ciphertext for attribute \( x \) along with \( M_0, M_1, \mathcal{A}' \) queries \( O_X(x) \rightarrow ( \{w_i\}_{x_i=1} ) \) in \( G_{\beta}^{1-\text{ABE}} \). \( \mathcal{A}' \) then samples \( c, c_i \leftarrow \mathbb{Z}_p^{2k+1} \) and \( b \leftarrow \{0,1\} \) (the challenge bit in the standard ABE security game) and returns
the following (SF) challenge ciphertext for \( A \):

\[
\mathbf{ct}_{x} := \left( [c^{T}]_{1}, \{ [c^{T}]_{1} \left( W + \hat{V}_{i}^{(2)} T + a^{\perp} w_{j} \right) + c^{T}_{i} W_{0} + \hat{V}_{i}^{(2)} U_{i} \}_{1}, \{ [c^{T}]_{1} \}_{x_{i}=1}, \right)
\]

\[
e([c^{T}]_{1}, \hat{V} + \hat{\mu}^{(0)} a^{\perp}) \cdot M_{b}
\]

- For the first \( \ell - 1 \) secret keys requested, say for formula \( f \), \( A' \) computes

\[
(\{v_{j}\}, \rho) \leftarrow \text{share}(f, \hat{V} + \hat{\delta} a^{\perp})
\]

where \( \hat{\delta} \leftarrow \mathbb{Z}_{p} \) is drawn independently for each key (here, the per-key \( \delta = \hat{\delta} - \mu^{(0)} \) implicitly). Next, for each \( j \), it queries \( O_{E}(\rho(j), [0]_{2}) \rightarrow ([w_{\rho(j)}^{T} r_{j}]_{2}, [r_{j}]_{2}) \) in \( G_{\beta}^{k-\text{ARE}} \) (since \( O_{E}(\rho(j), [0]_{2}) = \text{CPA. Enc}_{w_{\rho(j)}([0]_{2})} \)) and forms the following (SF) key:

\[
\text{sk}_{f} := \left( \{ [v_{j} + (W + \hat{V}_{\rho(j)}^{(2)} r_{j} + a^{\perp} w_{\rho(j)}^{T} r_{j}]_{2}, [r_{j}]_{2}, \right.
\]

\[
\left. ([W_{0} + \rho(j) \cdot W_{1} + U_{\rho(j)}^{(2)} r_{j}]_{2})_{\rho(j) \neq 0} \{ [v_{j}]_{2}\}_{\rho(j) = 0} \right)
\]

- For the last \( Q - \ell \) secret keys requested, say for formula \( f \), \( A' \) proceeds as before for the first \( \ell - 1 \) keys except

\[
(\{v_{j}\}, \rho) \leftarrow \text{share}(f, \hat{V} + \mu^{(0)} a^{\perp})
\]

It is easy to see that it forms a P-SF key.

- For the \( \ell \)th secret key requested, say for formula \( f \), \( A' \) computes \((\{v_{j}\}, \rho) \leftarrow \)


share(f, \bar{v}), queries \mathcal{O}_f(f) \rightarrow (\{[\mu_j + w^\top_{\rho(j)} r_j]_2, [r_j]_2\}) in G^{1-ABE}_\beta, then uses these components to return:

$$\text{sk}_f := \begin{cases} \{[v_j + (W + \bar{v}^{(2)}_{\rho(j)})r_j + a^\perp (\mu_j + w^\top_{\rho(j)} r_j)]_2, [r_j]_2, \}^{(v_j + a^\perp) + \bar{v}_{\rho(j)} r_j} \} \rho(j) \neq 0, \{[v_j]_2\} \rho(j) = 0, \end{cases}$$

We claim that if \beta = 0, then \text{sk}_f is a P-SF key, and if \beta = 1, then \text{sk}_f is a SF key. This follows the fact that thanks to linearity, the shares

$$\{v_j + \mu_j a^\perp, \rho\}, (\{v_j\}, \rho) \leftarrow \text{share}(f, \bar{v}), (\{\mu_j\}, \rho) \leftarrow \text{share}(f, \mu^{(\beta)})$$

are identically distributed to \text{share}(f, \bar{v} + \mu^{(\beta)} a^\perp). The claim then follows from the fact that \bar{v} + \mu^{(0)} a^\perp = v and that \bar{v} + \mu^{(1)} a^\perp is identically distributed to v + \delta a^\perp (where \delta = \mu^{(1)} - \mu^{(0)} is a fresh random value for this key).

Putting everything together, for \beta \in \{0, 1\}, when \mathcal{A}' interacts with G^{1-ABE}_{\beta}, then \mathcal{A}' simulates H_{2,\ell-1+\beta}. It follows then that:

$$|\Pr[\langle \mathcal{A}, H_{2,\ell-1}\rangle = 1] - \Pr[\langle \mathcal{A}, H_{2,\ell}\rangle = 1]| \leq |\Pr[\langle \mathcal{A}', G^{1-ABE}_{0}\rangle = 1] - \Pr[\langle \mathcal{A}', G^{1-ABE}_{1}\rangle = 1]|$$

From Theorem 18, we then have:

$$|\Pr[\langle \mathcal{A}, H_{2,\ell-1}\rangle = 1] - \Pr[\langle \mathcal{A}, H_{2,\ell}\rangle = 1]| \leq 2^{6d} \cdot 8^d \cdot n \cdot \text{Adv}^{k-LIN}_{G_{\beta}}(\lambda)$$
Lemma 32 \((H_{2,Q} \approx_s H_3)\).

\[
| \Pr[\langle \mathcal{A}, H_{2,Q} \rangle = 1] - \Pr[\langle \mathcal{A}, H_3 \rangle = 1] | \leq \frac{1}{p}
\]

Proof. These two hybrids are identically distributed conditioned on \(c^T a^\perp \neq 0\). To see this, consider two ways of sampling \(v\): as \(\tilde{v} \leftarrow \mathbb{Z}_p^{2k+1}\) and as \(\tilde{v} + \tilde{m} a^\perp\) for an independent \(\tilde{m} \leftarrow \mathbb{Z}_p\). Note that both result in \(v\) having a uniform distribution.

Using \(\tilde{v}\) to simulate hybrid \(H_{2,Q}\) obviously results in \(H_{2,Q}\) (where \(v = \tilde{v}\)). However, using the identically distributed \(v = \tilde{v} + \tilde{m} a^\perp\) to simulate \(H_{2,Q}\) results in \(H_3\) (where \(\tilde{M} = M \cdot e([c^T], [\tilde{m} a^\perp])\) is randomly distributed as long as \(c^T a^\perp \neq 0\), and for redefined independently random \(\tilde{\delta}_i := \delta_i + \tilde{m}\) in the secret keys).

\(c\) is chosen at random and independent from \(a^\perp \neq 0\), so \(c^T a^\perp = 0\) with probability \(\frac{1}{p}\), and since we know that \(H_{2,Q} \equiv H_3\) conditioned on \(c^T a^\perp \neq 0\), then we have:

\[
| \Pr[\langle \mathcal{A}, H_{2,Q} \rangle = 1] - \Pr[\langle \mathcal{A}, H_3 \rangle = 1] | \leq \frac{1}{p}
\]

\(\square\)

Theorem 33 (adaptive unbounded KP-ABE). The unbounded KP-ABE construction in Appendix 5.1 is adaptively secure under the MDDH\(k\) assumption.
Proof.

\[
|\Pr[\langle A, H_0 \rangle = 1] - \Pr[\langle A, H_3 \rangle = 1]| \leq |\Pr[\langle A, H_0 \rangle = 1] - \Pr[\langle A, H_1 \rangle = 1]| \\
+ \sum_{\ell=1}^{Q} |\Pr[\langle A, H_{2,\ell-1} \rangle = 1] - \Pr[\langle A, H_{2,\ell} \rangle = 1]| \\
+ |\Pr[\langle A, H_{2,Q} \rangle = 1] - \Pr[\langle A, H_3 \rangle = 1]| 
\]

(Since \( H_1 \equiv H_{2,0} \)). Summing the results of Lemmas 30, 31, and 32, we then have:

\[
|\Pr[\langle A, H_0 \rangle = 1] - \Pr[\langle A, H_3 \rangle = 1]| \leq (5n+1) \cdot \text{Adv}_{G_r}^{k-\text{LIN}}(\lambda) + Q \cdot 2^d \cdot 8^d \cdot n \cdot \text{Adv}_{G^*}^{k-\text{LIN}}(\lambda) + \frac{1}{p}
\]

If \( d = O(\log n) \), then under the \( k \)-Lin Assumption this is a negligible function of \( \lambda \) (the number of queries made \( Q \) and the attribute vector length \( n \) are both polynomial in \( \lambda \), and \( \frac{1}{p} \) is a negligible function of \( \lambda \)). It’s easy to see that \( \text{Adv}_{A}^{\text{ABE}}(\lambda) = 0 \) in the \( H_3 \) hybrid game (since a random message is encrypted in the challenge ciphertext). So, any adversary in the real game \( (H_0) \) will have advantage negligibly close to 0, and our construction satisfies adaptive security.

\[ \square \]
Conclusion

5.2 Summary of Results

To conclude, in this work we presented the first ABE schemes for Boolean formulas that simultaneously:

(1) achieve adaptive security (with polynomial security loss);
(2) rely on a simple static hardness assumption in the standard model; and
(3) enjoy a compactness property, where the size of either the key or ciphertext grows only with the size of the attribute vector and is independent of the complexity of the supported policy class (even for complex policies that refer to each attribute many times).

We give both Key-Policy and Ciphertext-Policy constructions with these properties in Sections 4.5 and 4.6 respectively.

A key technical result used to prove the adaptive security of these constructions (Theorem 18) answers the open question of [Jaf+17], namely, proving the adaptive security of Yao Secret Sharing for \( \text{NC}^1 \) circuits with a polynomial (rather than subexponential) security loss, as discussed in Section 4.4.

Along the way, we developed a modular theory of unboundedness for ABE con-
structions, resulting in the first ABE construction for arithmetic span programs that achieves adaptive security under a static assumption. The technical details of this theory are outlined in Section 3.3.

Finally, we demonstrated the utility of our modular theory by using it to immediately obtain an unbounded version of our KP-ABE construction for Boolean formulas in Section 5.1 (which is therefore the first unbounded compact ABE construction for Boolean formulas proved adaptively secure from a static assumption).

5.3 Future Directions

Although general Boolean formula access policies are quite versatile, one can imagine more expressive models of computation, like general Boolean circuits, for example. Currently, there are no constructions for attribute-based encryption for Boolean circuits that satisfy adaptive security under a simple assumption. The closest constructions known are those of [GVW13; BV16] which support Boolean circuit access policies, but are only proven selectively secure and semi-adaptively secure under the Learning With Errors (LWE) assumption. Boosting these constructions to achieve adaptive security is an interesting open problem. A natural approach would be to develop some kind of analogue of the standard dual system proof technique in the lattice regime.

Also, we note that we do not even have a direct candidate construction of ABE for Boolean circuits in the bilinear group regime (even in the generic group model). Another interesting open problem would be to develop such a construction or prove
that a construction is impossible.
Bibliography


