Compact Adaptively Secure ABE for NC^1 from k-Lin

Lucas Kowalczyk[1,⋆] and Hoeteck Wee[2,⋆⋆]

[1] Columbia University
luke@cs.columbia.edu

[2] CNRS, ENS, PSL
wee@di.ens.fr

Abstract. We present compact attribute-based encryption (ABE) schemes for NC^1 that are adaptively secure under the k-Lin assumption with polynomial security loss. Our KP-ABE scheme achieves ciphertext size that is linear in the atttribute length and independent of the policy size even in the many-use setting, and we achieve an analogous efficiency guarantee for CP-ABE. This resolves the central open problem posed by Lewko and Waters (CRYPTO 2011). Previous adaptively secure constructions either impose an attribute "one-use restriction" (or the ciphertext size grows with the policy size), or require q-type assumptions.

1 Introduction

Attribute-based encryption (ABE) [31,17] is a generalization of public-key encryption to support fine-grained access control for encrypted data. Here, ciphertexts and keys are associated with descriptive values which determine whether decryption is possible. In a key-policy ABE (KP-ABE) scheme for instance, ciphertexts are associated with attributes like '(author:Waters), (inst:UT), (topic:PK)' and keys with access policies like ((topic:MPC) OR (topic:SK)) AND (NOT(inst:UCL)), and decryption is possible only when the attributes satisfy the access policy. A ciphertext-policy (CP-ABE) scheme is the dual of KP-ABE with ciphertexts associated with policies and keys with attributes.

Over past decade, substantial progress has been made in the design and analysis of ABE schemes, leading to a large families of schemes that achieve various trade-offs between efficiency, security and underlying assumptions. Meanwhile,

⋆ Supported in part by an NSF Graduate Research Fellowship DGE-16-44869; The Leona M. & Harry B. Helmsley Charitable Trust; ERC Project aSCEND (H2020 639554); the Defense Advanced Research Project Agency (DARPA) and Army Research Office (ARO) under Contract W911NF-15-C-0236; and NSF grants CNS-1445424, CNS-1552932 and CCF-1423306. Any opinions, findings and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the the Defense Advanced Research Projects Agency, Army Research Office, the National Science Foundation, or the U.S. Government.

⋆⋆ Supported in part by ERC Project aSCEND (H2020 639554).

ABE has found use as a tool for providing and enhancing privacy in a variety of settings from electronic medical records to messaging systems and online social networks. Moreover, we expect further deployment of ABE, thanks to the recent standardization efforts of the European Telecommunications Standards Institute (ETSI).

In this work, we consider KP-ABE schemes for access policies in NC^1 that simultaneously:

(1) enjoy compact ciphertexts whose size grows only with the length of the attribute and is independent of the policy size, even for complex policies that refer to each attribute many times;
(2) achieve adaptive security (with polynomial security loss);
(3) rely on simple hardness assumptions in the standard model;
(4) can be built with asymmetric prime-order bilinear groups.

We also consider the analogous question for CP-ABE schemes with compact keys. In both KP and CP-ABE, all four properties are highly desirable from both a practical and theoretical stand-point and moreover, properties (1), (2) and (4) are crucial for many real-world applications of ABE. In addition, properties (2), (3) and (4) are by now standard cryptographic requirements pertaining to speed and efficiency, strong security guarantees under realistic and natural attack models, and minimal hardness assumptions. There is now a vast body of works on ABE (e.g. [17,23,27,26], see Fig 1) showing how different combinations of (1) – (4), culminating in several unifying frameworks that provide a solid understanding of the design and analysis of these schemes [2,34,6,3,1]. Nonetheless, prior to this work, it was not known how to even simultaneously realize (1) – (3) for NC^1 access policies[3]; indeed, this is widely regarded one of the main open problems in pairing-based ABE.

Our results. We present the first KP-ABE and CP-ABE schemes for NC^1 that simultaneously realize properties (1) – (4). Our KP-ABE scheme achieves ciphertext size that is linear in the attribute 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 with polynomial security loss. We also present an "unbounded" variant of our compact KP-ABE scheme with constant-size public parameters.

As an immediate corollary, we obtain delegation schemes for NC^1 with public verifiability and adaptive soundness under the k-Lin assumption [30,26,9].

Our construction leverages a refinement of the recent "partial selectivization" framework for adaptive security [20] (which in turn builds upon [13,12,18,21]) along with the classic dual system encryption methodology [33,26].

[3] Note that there exist constructions of ABE for more general access policies like monotone span programs / Boolean formulas with threshold gates [17], and even polynomial-sized Boolean circuits [16,14], but all such constructions sacrifice at least one of the properties (1)-(3).

reference	adaptive	compact	assumption	unbounded
GPSW [17]		✓	static ✓	
LOSTW [23,27]	✓		static ✓	
LW [26]	✓	✓	q-type	
OT [28]	✓		2-Lin ✓	✓
Att [3]	✓	✓	q-type	✓
CGKW [7]	✓		k-Lin ✓	✓
ours, Section 6	✓	✓	static ✓	
ours, Section **??**	✓	✓	static ✓	✓

Fig. 1. Summary of KP-ABE schemes for NC1

1.1 Technical overview

Our starting point is the Lewko-Waters framework for constructing compact adaptively secure ABE [26] based on the dual system encryption methodology[4] [33,25,23]. Throughout, we focus on monotone NC^1 circuit access policies, and note that the constructions extend readily to the non-monotone setting[5]. 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.

Warm-up. We begin with the prior compact KP-ABE for monotone NC^1 [26,23,17]; this is an adaptively secure scheme that comes with the downside of relying on q-type assumptions (q-type assumptions are assumptions of size that grows with some parameter q. It is known that many q-type assumptions become stronger as q grows [10], and in general such complex and dynamic assumptions are not well-understood). The construction uses composite-order groups, but here we'll suppress the distinction between composite-order and prime-order groups for simplicity. We associate ciphertexts $\mathsf{ct}_{\mathbf{x}}$ with attribute vectors[6] $\mathbf{x} \in \{0,1\}^n$

[4] Essentially, the dual system proof method provides guidance for transforming suitably-designed functional encryption schemes which are secure for one adversarial secret key request to the multi-key setting where multiple keys may be requested by the adversary. Our main technical contribution involves the analysis of the initial single-key-secure component, which we refer to later as our "Core 1-ABE" component.

[5] Most 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 [29,24], but we leave this for future work.

[6] Some works associate ciphertexts with a set $S \subseteq [n]$ where $[n]$ is referred to as the attribute universe, in which case $\mathbf{x} \in \{0,1\}^n$ corresponds to the characteristic vector of S.

and keys sk_f with Boolean formulas f:

$$\begin{aligned} \mathsf{msk} &:= (\mu, w_1, \ldots, w_n) \\ \mathsf{mpk} &:= (g, g^{w_1}, \ldots, g^{w_n}, e(g,h)^{\mu}), \\ \mathsf{ct}_{\mathbf{x}} &:= (g^s, \{g^{s w_i}\}_{x_i=1}, e(g,h)^{\mu s} \cdot M) \\ \mathsf{sk}_f &:= (\{h^{\mu_j + r_j w_{\rho(j)}}, h^{r_j}\}_{j \in [m]}), \rho : [m] \to [n] \end{aligned} \tag{1}$$

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 $\mathbf{x} \in \{0,1\}^n$, the shares $\{\mu_j\}_{x_{\rho(j)}=1}$ determine μ if $\mathbf{x}$ satisfies f (i.e., $f(\mathbf{x}) = 1$), and reveal nothing about μ otherwise; and ρ 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 $m = \mathrm{poly}(n) \gg n$. 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 μ remains computationally hidden given just a single ciphertext and a single key and no mpk (the more general setting with mpk and multiple keys follows via what is by now a textbook application of the dual system encryption methodology). In fact, it suffices to show that μ is hidden given just

$$\begin{aligned} \mathsf{ct}'_{\mathbf{x}} &:= (\{w_i\}_{x_i=1}) \quad // \text{ "stripped down" } \mathsf{ct}_{\mathbf{x}} \\ \mathsf{sk}_f &:= (\{h^{\mu_j + r_j w_{\rho(j)}}, h^{r_j}\}_{j \in [m]}) \end{aligned}$$

where $\mathbf{x}, f$ are adaptively chosen subject to the constraint $f(\mathbf{x}) = 0$. Henceforth, we refer to $(\mathsf{ct}'_{\mathbf{x}}, \mathsf{sk}_f)$ as our "core 1-ABE component". Looking ahead to our formalization of adaptive security for this core 1-ABE, we actually require that μ 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 [26,4,2,1], or imposes the one-use restriction (that is, ρ is injective and $m = n$, in which case security can be achieved unconditionally) [23,34]. 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 [20] (which in turn builds upon [13,12,18,21]).

Let G_0 denote the view of the adversary $(\mathsf{ct}'_{\mathbf{x}}, \mathsf{sk}_f)$ in the real game, and G_1 denote the same thing except we replace $\{\mu_j\}$ in sk_f with shares of a random value independent of μ. Our goal is to show that $\mathsf{G}_0 \approx_c \mathsf{G}_1$. First, let us define an additional family of games $\{\mathsf{H}^U\}$ 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, $\mathsf{H}^{\emptyset} = \mathsf{G}_0$.

We begin with the "selective" setting, where the adversary specifies $\mathbf{x}$ at the start of the game. Suppose we can show that $\mathsf{G}_0 \approx_c \mathsf{G}_1$ in this simpler setting via a series of $L+1$ hybrids of the form:

$$\mathsf{G}_0 = \mathsf{H}^{h_0(\mathbf{x})} \approx_c \mathsf{H}^{h_1(\mathbf{x})} \approx_c \cdots \approx_c \mathsf{H}^{h_L(\mathbf{x})} = \mathsf{G}_1$$

where $h_0, \ldots, h_L : \{0,1\}^n \to \{U \subseteq [m] : |U| \leq R'\}$ are functions of the adversary's choices $\mathbf{x}$. Then, the piecewise guessing framework basically tells us that $\mathsf{G}_0 \approx_c \mathsf{G}_1$ 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 $h_i(\mathbf{x})$ (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 $\mathbf{x}$ itself

First, we describe a straight-forward approach which achieves $L = 2$ and $R' = m$ implicit in [26] (but incurs a huge security loss $2^m \gg 2^n$) where

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

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

- $\mathsf{G}_0 \approx_c \mathsf{H}^{h_1(\mathbf{x})}$ via DDH, since $h^{\mu_j + w_{\rho(j)} r_j}, h^{r_j}$ computationally hides μ_j whenever $x_{\rho(j)} = 0$ and $w_{\rho(j)}$ is not leaked in $\mathsf{ct}_{\mathbf{x}}$;
- $\mathsf{H}^{h_1(\mathbf{x})} \approx_s \mathsf{G}_1$ via security of the secret-sharing scheme since the shares $\{\mu_j : x_{\rho(j)} = 1\}$ leak no information about μ whenever $f(\mathbf{x}) = 0$.

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 NC^1 in [20] (which is in turn a variant of Yao's secret-sharing scheme [32,19]), 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 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 $2^{O(d)} = \mathrm{poly}(n)$.

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. [6,15,5] 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 (1) and carry out the following substitutions:

$$g^s \mapsto [\mathbf{s}^\top \mathbf{A}]_1,\ h^{r_j} \mapsto [\mathbf{r}_j]_2,\ w_i \mapsto \mathbf{W}_i \leftarrow \mathbb{Z}_p^{(k+1)\times k},\ \mu \mapsto \mathbf{v} \leftarrow \mathbb{Z}_p^{k+1} \tag{2}$$

where $\mathbf{A} \leftarrow \mathbb{Z}_p^{k\times(k+1)}, \mathbf{s}, \mathbf{r}_j \leftarrow \mathbb{Z}_p^k$, k corresponds to the k-Lin Assumption desired for security[7], and $[\cdot]_1, [\cdot]_2$ correspond respectively to exponentiations in the prime-order groups G_1, G_2. We note that the naive transformation following [6] would have required $\mathbf{W}_i$ of dimensions at least $(k+1)\times(k+1)$; here, we incorporated optimizations from [15,5]. This yields the following prime-order KP-ABE scheme for NC^1:

$$\begin{aligned}
\mathsf{msk} &:= (\ \mathbf{v}, \mathbf{W}_1, \ldots, \mathbf{W}_n\ ) \\
\mathsf{mpk} &:= (\ [\mathbf{A}]_1, [\mathbf{A}\mathbf{W}_1]_1, \ldots, [\mathbf{A}\mathbf{W}_n]_1,\ e([\mathbf{A}]_1, [\mathbf{v}]_2)\ ), \\
\mathsf{ct}_{\mathbf{x}} &:= \left(\ [\mathbf{s}^\top \mathbf{A}]_1, \{[\mathbf{s}^\top \mathbf{A}\mathbf{W}_i]_1\}_{x_i=1},\quad e([\mathbf{s}^\top \mathbf{A}]_1, [\mathbf{v}]_2)\cdot M\ \right) \\
\mathsf{sk}_f &:= (\ \{[\mathbf{v}_j + \mathbf{W}_{\rho(j)}\mathbf{r}_j]_2, [\mathbf{r}_j]_2\}_{j\in[m]}\ )
\end{aligned}$$

where $\mathbf{v}_j$ is the j'th share of $\mathbf{v}$. Decryption proceeds as before by first computing

$$\{e([\mathbf{s}^\top \mathbf{A}]_1, [\mathbf{v}_j]_2)\}_{\rho(j)=0 \vee x_{\rho(j)}=1}$$

and relies on the associativity relations $\mathbf{A}\mathbf{W}_i \cdot \mathbf{r}_j = \mathbf{A} \cdot \mathbf{W}_i \mathbf{r}_j$ for all i, j [8].

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

$$(\mathbf{A}, \mathbf{A}\mathbf{W}_i, [\mathbf{W}_i \mathbf{r}_j]_2, [\mathbf{r}_j]_2) \approx_c (\mathbf{A}, \mathbf{A}\mathbf{W}_i, [\mathbf{W}_i \mathbf{r}_j + \delta_{ij}\mathbf{a}^\perp]_2, [\mathbf{r}_j]_2)$$

where $\mathbf{A} \leftarrow \mathbb{Z}_p^{k\times(k+1)}, \mathbf{W}_i \leftarrow \mathbb{Z}_p^{(k+1)\times 2k}, \mathbf{r}_j \leftarrow \mathbb{Z}_p^{2k}$ and $\mathbf{a}^\perp \in \mathbb{Z}_p^{k+1}$ satisfies $\mathbf{A} \cdot \mathbf{a}^\perp = \mathbf{0}$.

Organization. We describe the piecewise guessing framework for adaptive security in Section 3 and a pebbling strategy (used to define $h_0, \ldots, h_L$) in Section 4. We describe a secret-sharing scheme and prove adaptive security of the core 1-ABE component in Section 5. We present our full KP-ABE scheme in Section 6, and present a CP-ABE and unbounded KP-ABE scheme in the full version of this paper [22].

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.

[7] 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).

Throughout this paper, we use 1^λ 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 formula. A monotone Boolean formula $f : \{0,1\}^n \to \{0,1\}$ 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 $g : \{0,1\}^2 \to \{0,1\}$ 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 formula.* 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 NC^1 circuits, so the blowup from this procedure is polynomial in n. Hence we will consider the class 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 $f \in \mathsf{NC}^1$ formulas.

2.2 Secret sharing

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

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

$$\mathsf{reconstruct}(f, x, \{\mu_j\}_{\rho(j)=0 \vee x_{\rho(j)}=1}) = \mu.$$

– Security stipulates that for every $x \in \{0,1\}^n$ such that $f(x) = 0$, the shares $\{\mu_j\}_{\rho(j)=0 \vee x_{\rho(j)}=1}$ perfectly hide μ.

Note the inclusion of $\rho(j) = 0$ in both correctness and security. All the secret sharing schemes in this work will in fact be linear (in the standard sense): share computes a linear function of the secret μ and randomness over $\mathbb{Z}_p$, and $\mathsf{reconstruct}$ computes a linear function of the shares over $\mathbb{Z}_p$, that is, $\mu = \sum_{\rho(j)=0 \vee x_{\rho(j)=1}} \omega_j \mu_j$.

2.3 Attribute-based encryption

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

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

$\mathsf{Enc}(\mathsf{mpk}, x, m) \to \mathsf{ct}_x$. The encryption algorithm gets as input mpk, an attribute $x \in \mathcal{X}$ and a message $m \in \mathcal{M}$. It outputs a ciphertext ct_x. Note that x is public given ct_x.

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, y) \to \mathsf{sk}_y$. The key generation algorithm gets as input msk and a value $y \in \mathcal{Y}$. It outputs a secret key sk_y. Note that y is public given sk_y.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_y, \mathsf{ct}_x) \to m$. The decryption algorithm gets as input sk_y and ct_x such that $\mathsf{pred}(x, y) = 1$. It outputs a message m.

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

$$\Pr[\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_y, \mathsf{Enc}(\mathsf{mpk}, x, m)) = m] = 1,$$

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

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

$$\mathsf{Adv}^{\mathrm{ABE}}_{\mathcal{A}}(\lambda) := \Pr\left[b = b' : \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}, \mathcal{M}); \\ (x^*, m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{mpk}); \\ b \leftarrow_{\mathrm{R}} \{0,1\}; \mathsf{ct}_{x^*} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x^*, m_b); \\ b' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{ct}_{x^*}) \end{array}\right] - \frac{1}{2}$$

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

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

A generator $\mathcal{G}$ takes as input a security parameter λ and outputs a group description $\mathbb{G} := (p, G_1, G_2, G_T, e)$, where p is a prime of $\Theta(\lambda)$ bits, G_1, G_2 and G_T are cyclic groups of order p, and $e : G_1 \times G_2 \to G_T$ 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 λ. Let $g_1 \in G_1$, $g_2 \in G_2$ and $g_T = e(g_1, g_2) \in G_T$ be the respective generators. We employ the *implicit representation* of group elements: for a matrix $\mathbf{M}$ over $\mathbb{Z}_p$, we define $[\mathbf{M}]_1 := g_1^{\mathbf{M}}, [\mathbf{M}]_2 := g_2^{\mathbf{M}}, [\mathbf{M}]_T := g_T^{\mathbf{M}}$, where exponentiation is carried out component-wise. Also, given $[\mathbf{A}]_1, [\mathbf{B}]_2$, we let $e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{AB}]_T$.

We define the matrix Diffie-Hellman (MDDH) assumption on G_1 [11]:

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

$$\mathsf{Adv}^{\mathrm{MDDH}^m_{k,\ell}}_{\mathcal{A}}(\lambda) := \left| \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{MS}]_1) = 1] - \Pr[\mathcal{A}(\mathbb{G}, [\mathbf{M}]_1, [\mathbf{U}]_1) = 1] \right|$$

where $\mathbf{M} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{\ell \times k}$, $\mathbf{S} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{k \times m}$ and $\mathbf{U} \leftarrow_{\mathrm{R}} \mathbb{Z}_p^{\ell \times m}$.

The MDDH assumption on G_2 can be defined in an analogous way. Escala *et al.* [11] showed that

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

with a tight security reduction (that is, $\mathsf{Adv}^{\mathrm{MDDH}^m_{k,\ell}}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{k\text{-Lin}}_{\mathcal{A}'}(\lambda)$). In fact, the MDDH assumption is a generalization of the k-Lin Assumption, such that the k-Lin Assumption is equivalent to the $\mathrm{MDDH}^1_{k,k+1}$ Assumption as defined above.

Definition 2 (k-Lin Assumption). *Let $k \geq 1$. We say that the k-Lin Assumption holds if for all PPT adversaries $\mathcal{A}$, the following advantage function is negligible in λ.*

$$\mathsf{Adv}^{k\text{-Lin}}_{\mathcal{A}}(\lambda) := \mathsf{Adv}^{\mathrm{MDDH}^1_{k,k+1}}_{\mathcal{A}}(\lambda)$$

Henceforth, we will use MDDH_k to denote $\mathrm{MDDH}^1_{k,k+1}$. Lastly, we note that the k-Lin Assumption itself is a generalization, where setting $k = 1$ yields the Symmetric External Diffie-Hellman Assumption (SXDH), and setting $k = 2$ yields the standard Decisional Linear Assumption (DLIN).

3 Piecewise Guessing Framework for Adaptive Security

We now refine the adaptive security framework of [20], making some simplifications along the way to yield the piecewise guessing framework that will support our security proof. We use $\langle \mathsf{A}, \mathsf{G} \rangle$ 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 $\Pr[\langle \mathsf{A}, \mathsf{G} \rangle = 1]$.

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 $\mathsf{G}_0, \mathsf{G}_1$ are ϵ-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 $\mathsf{G}_0, \mathsf{G}_1$ 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 $\mathsf{G}_0 \approx_c \mathsf{G}_1$ in this simpler setting via a series of $L+1$ hybrids of the form:

$$\mathsf{G}_0 = \mathsf{H}^{h_0(z^*)} \approx_c \mathsf{H}^{h_1(z^*)} \approx_c \cdots \approx_c \mathsf{H}^{h_L(z^*)} = \mathsf{G}_1$$

where $h_0, \ldots, h_L : \{0,1\}^R \to \{0,1\}^{R'}$ and $\{\mathsf{H}^u\}_{u \in \{0,1\}^{R'}}$ is a family of games where the messages sent to the adversary in H^u depend on u.[8] In particular, the ℓ'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 $\mathsf{G}_0 \approx_c \mathsf{G}_1$ even for an adaptive choice of z. Note that $\{\mathsf{H}^u\}_{u \in \{0,1\}^{R'}}$ is now a family of adaptive games where z is adaptively defined as the game progresses. We have two requirements:

The first, *end-point equivalence*, just says the two equivalences

$$\mathsf{G}_0 = \mathsf{H}^{h_0(z^*)},\ \mathsf{G}_1 = \mathsf{H}^{h_L(z^*)}$$

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

$$\mathsf{H}^{u_0} \approx_c \mathsf{H}^{u_1},\ \forall\, u_0, u_1 \in \{0,1\}^{R'}$$

as long as the adversary chooses z such that $h_{\ell-1}(z) = u_0 \wedge h_\ell(z) = u_1$ It is easy to see that this is a generalization of $\mathsf{H}^{h_{\ell-1}(z^*)} \approx_c \mathsf{H}^{h_\ell(z^*)}$ 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 $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1), \widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)$ for any $u_0, u_1 \in \{0,1\}^{R'}$, where

[8] Informally, $\{\mathsf{H}^u\}$ 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 ℓ'th hybrid. In the ℓ'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 commited to u, but the adversary can do whatever it wants.

$\widehat{\mathsf{H}}_{\ell,b}(u_0, u_1)$ 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 $(h_{\ell-1}(z), h_\ell(z)) = (u_0, u_1)$.

Adaptive security lemma. The next lemma tells us that the two requirements above implies that $\mathsf{G}_0 \approx_c \mathsf{G}_1$ with a security loss $2^{2R'} \cdot L$ (stated in the contrapositive). In our applications, $2^{R'}$ and L will be polynomial in the security parameter.

Lemma 1 (adaptive security lemma). *Fix $\mathsf{G}_0, \mathsf{G}_1$ along with $h_0, h_1, \ldots, h_L : \{0,1\}^R \to \{0,1\}^{R'}$ and $\{\mathsf{H}^u\}_{u \in \{0,1\}^{R'}}$ such that*

$$\forall\, z^* \in \{0,1\}^R : \mathsf{H}^{h_0(z^*)} = \mathsf{G}_0,\ \mathsf{H}^{h_L(z^*)} = \mathsf{G}_1$$

Suppose there exists an adversary A such that $\Pr[\langle \mathsf{A}, \mathsf{G}_0 \rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{G}_1 \rangle = 1] \geq \epsilon$ then there exists $\ell \in [L]$ and $u_0, u_1 \in \{0,1\}^{R'}$ such that

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

This lemma is essentially a restatement of the main theorem of [20, Theorem 2]; we defer a comparison to the end of this section.

Proof. For the proof, we need to define the game $\mathsf{H}_\ell(z^*)$ for all $\ell = 0, 1, \ldots, L$ and all $z^* \in \{0,1\}^R$

$\mathsf{H}_\ell(z^*)$ is the same as $\mathsf{H}^{h_\ell(z^*)}$, except we replace the output with 0 whenever $z \neq z^*$.

Roughly speaking, in $\mathsf{H}_\ell(z^*)$, the adversary "commits" to making choices $z = z^*$ in advance.

– Step 1. We begin the proof by using "random guessing" to deduce that

$$\Pr_{z^* \leftarrow \{0,1\}^R}[\langle \mathsf{A}, \mathsf{H}_0(z^*) \rangle = 1] - \Pr_{z^* \leftarrow \{0,1\}^R}[\langle \mathsf{A}, \mathsf{H}_L(z^*) \rangle = 1] \geq \frac{\epsilon}{2^R}$$

This follows from the fact that $\mathsf{H}^{h_0(z)} = \mathsf{G}_0, \mathsf{H}^{h_L(z)} = \mathsf{G}_1$ which implies

$$\Pr_{z^* \leftarrow \{0,1\}^R}[\langle \mathsf{A}, \mathsf{H}_0(z^*) \rangle = 1] = \frac{1}{2^R} \Pr[\langle \mathsf{A}, \mathsf{G}_0 \rangle = 1]$$
$$\Pr_{z^* \leftarrow \{0,1\}^R}[\langle \mathsf{A}, \mathsf{H}_L(z^*) \rangle = 1] = \frac{1}{2^R} \Pr[\langle \mathsf{A}, \mathsf{G}_1 \rangle = 1].$$

– Step 2. Via a standard hybrid argument, we have that there exists ℓ such that

$$\Pr_{z^* \leftarrow \{0,1\}^R}[\langle \mathsf{A}, \mathsf{H}_{\ell-1}(z^*)\rangle = 1] - \Pr_{z^* \leftarrow \{0,1\}^R}[\langle \mathsf{A}, \mathsf{H}_{\ell}(z^*)\rangle = 1] \geq \frac{\epsilon}{2^R L}$$

which implies that:

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

– Step 3. Next, we relate $\widehat{\mathsf{H}}_{\ell,0}, \widehat{\mathsf{H}}_{\ell,1}$ and $\mathsf{H}_{\ell-1}, \mathsf{H}_{\ell}$. 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

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

Then, we have

$$\begin{aligned}
&\sum_{z' \in \{0,1\}^R} \Pr[\langle \mathsf{A}, \mathsf{H}_{\ell}(z')\rangle = 1] \\
&= \sum_{(u_0,u_1) \in \mathcal{U}_\ell} \left(\sum_{z':(h_{\ell-1}(z'),h_\ell(z'))=(u_0,u_1)} \Pr[\langle \mathsf{A}, \mathsf{H}_{\ell}(z')\rangle = 1] \right) \\
&= \sum_{(u_0,u_1) \in \mathcal{U}_\ell} \Pr[\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,1}(u_0,u_1)\rangle = 1]
\end{aligned}$$

By the same reasoning, we also have

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

This means that

$$\begin{aligned}
&\sum_{(u_0,u_1) \in \mathcal{U}_\ell} \Big(\Pr[\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,0}(u_0,u_1)\rangle = 1] - \Pr[\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,1}(u_0,u_1)\rangle = 1] \Big) \\
&= \sum_{z' \in \{0,1\}^R} \Pr[\langle \mathsf{A}, \mathsf{H}_{\ell-1}(z')\rangle = 1] - \sum_{z' \in \{0,1\}^R} \Pr[\langle \mathsf{A}, \mathsf{H}_{\ell}(z')\rangle = 1] \geq \frac{\epsilon}{L}
\end{aligned}$$

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

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

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

Comparison with [20]. Our piecewise guessing framework makes explicit the game H^u which are described implicitly in the applications of the framework in [20]. Starting from H^u and $h_0, \ldots, h_L$, we can generically specify the intermediate games $\widehat{\mathsf{H}}_{\ell,0}, \widehat{\mathsf{H}}_{\ell,1}$ as well as the games $\mathsf{H}_0, \ldots, \mathsf{H}_L$ used in the proof of security. The framework of [20] does the opposite: it starts with the games $\mathsf{H}_0, \ldots, \mathsf{H}_L$, and the theorem statement assumes the existence of $h_0, \ldots, h_L$ and $\widehat{\mathsf{H}}_{\ell,0}, \widehat{\mathsf{H}}_{\ell,1}$ that are "consistent" with $\mathsf{H}_0, \ldots, \mathsf{H}_L$ (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 $\widehat{\mathsf{H}}_{\ell,0}, \widehat{\mathsf{H}}_{\ell,1}$ and H_ℓ in the applications.

Finally, we point out that the sets $\mathcal{U}$ and $\mathcal{W}$ in [20, 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 ℓ'th function $h_{\ell-1} \circ h_\ell$ over here.

4 Pebbling Strategy for NC^1

We now define a pebbling strategy for 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 $f : \{0,1\}^n \to \{0,1\}$ 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 [20, Section 4]; the main difference is that we provide a better bound on the size of the description of each pebbling configuration in Theorem 1.

4.1 Pebbling Rules

Fix a formula $f : \{0,1\}^n \to \{0,1\}$ 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 3 (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 3.

4.2 Pebbling Strategy

Given an 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 2 and begins by calling $\mathsf{Pebble}(f, x, g^*)$ on the root gate g^*. We give an example in Figure 3.

$\mathsf{Pebble}(f, x, g)$:
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 $\mathsf{Pebble}(f, x, g_L)$ to get a list of operations Λ_L, then call $\mathsf{Pebble}(f, x, g_R)$ to get a second list of operations λ_R.
   Return $\Lambda_L \circ \Lambda_R \circ$ "PEBBLE g" $\circ\ \mathsf{Reverse}(\Lambda_R) \circ\ \mathsf{Reverse}(\Lambda_L)$
3. (Recursive Case) If $g = \text{AND}$, call $\mathsf{Pebble}(f, x, \cdot)$ on the first child gate whose output wire evaluates to 0 on input x to get a list of operations Λ.
   Return $\Lambda \circ$ "PEBBLE g" $\circ\ \mathsf{Reverse}(\Lambda)$

$\mathsf{Reverse}(\Lambda)$:
Input: A list of instructions of the form "PEBBLE g" or "unPEBBLE g" for a gate g.

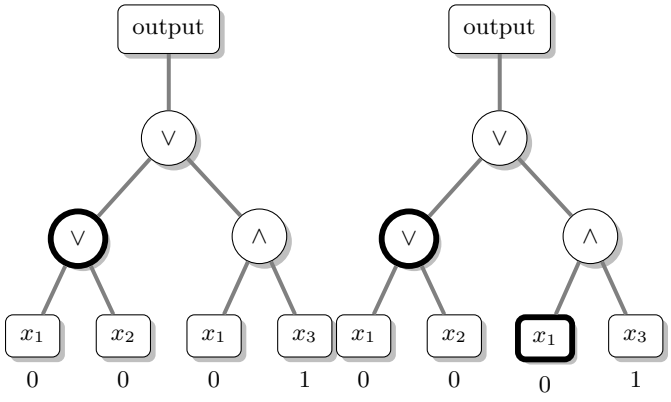
1. Return the list Λ in the reverse order, additionally changing each original "PEBBLE " instruction to "unPEBBLE " and each original "unPEBBLE " instruction to "PEBBLE ".

Fig. 2. NC^1 formula pebbling strategy.

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 $\mathsf{Pebble}()$ 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 $\mathsf{Pebble}()$ procedure is called will have child nodes with output wires which *both* evaluate to 0. Furthermore, by inspection, $\mathsf{Pebble}(f, x, g^*)$ returns a sequence of pebbling instructions for the circuit that follows the rules in Definition 3.

4.3 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



step	move
1	pebble x_1 (left)
2	pebble x_2
3	pebble $\vee$ (left)
4	unpebble x_2
5	unpebble x_1 (left)
6	pebble x_1 (right)
7	pebble $\wedge$
8	unpebble x_1 (right)
9	pebble $\vee$ (right)
10	pebble x_1 (right)
11	unpebble $\wedge$
12	unpebble x_1 (right)
13	pebble x_1 (left)
14	pebble x_2
15	unpebble $\vee$ (left)
16	unpebble x_2
17	unpebble x_1 (left)

Fig. 3. 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 $\vee$ node leading to the output node.

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 1 (pebbling NC^1). *For every input $x \in \{0,1\}^n$ and any monotone formula f of depth d and fan-in two for which $f(x) = 0$, there exists a sequence of $L(d) = 8^d$ pebbling instructions such that every intermediate pebbling configuration can be described using $R'(d) = 3d$ bits.*

Proof. Follows from the joint statements of Lemma 2 and Lemma 4 applied to the pebbling strategy in Figure 2.

Comparison with [20]. Note that the strategy reproduced in Figure 2 is essentially the same as one analyzed by [20], 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 4

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 3 and 4 show that every configuration produced follows a pattern that can be described using only $3d$ bits.

Lemma 2 ([20]). *The pebbling strategy in Figure 2 called on the root gate g^* for a formula f of depth d with assignment x such that $f(x) = 0$, $\mathsf{Pebble}(f, x, g^*)$, returns a sequence of instructions of length at most $L(d) \leq 8^d$.*

This bound is a special case of that shown in [20, 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.

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:

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

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 3 (structure of pebbling configuration). *Every configuration induced by application of the instructions produced by the pebbling strategy in Figure 2 called on the root gate g^* of a formula f of depth d with assignment x such that $f(x) = 0$, $\mathsf{Pebble}(f, x, g^*)$, 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 $\mathsf{Reverse}(L_1 \circ L_2)$ as $\mathsf{Reverse}(L_2) \circ \mathsf{Reverse}(L_1)$.

We proceed with our original proof via a proof by induction on the depth of the formula upon which $\mathsf{Pebble}()$ is called.

Inductive Hypothesis: For formulas f of depth $d-1$ with root gate g^* and assignment x such that $f(x) = 0$, $\mathsf{Pebble}(f, x, g^*)$ 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 $\mathsf{Pebble}(f, x, g^*)$ 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 $\mathsf{Pebble}(f, x, g^*)$ 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

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

(The case with g_L^* instead of g_R^* is handled analogously, even simpler). Suppose $\mathsf{Pebble}(f_{g_R^*}, x, g_R^*)$ (and thus $\mathsf{Reverse}(\mathsf{Pebble}(f_{g_R^*}, x, g_R^*))$) 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 $f_{g_L^*}$ 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 $\mathsf{Reverse}$ 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

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

Suppose $\mathsf{Pebble}(f_{g_R^*}, x, g_R^*)$, $\mathsf{Pebble}(f_{g_L^*}, x, g_L^*)$, and thus $\mathsf{Reverse}(\mathsf{Pebble}(f_{g_R^*}, x, g_R^*))$, $\mathsf{Reverse}(\mathsf{Pebble}(f_{g_L^*}, x, g_L^*))$, 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^*. 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^*, $f_{g_R^*}$, 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^*, so it is also good. Finally, by part (1) of this application of the inductive hypothesis, we know that L_0th configuration features a single pebble on g_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^* except for the single pebble on g_L^*. 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^* 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 $\mathsf{Reverse}$ 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 $\mathsf{Reverse}$ 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^*.

□

Lemma 4 ($R'(d) = 3d$)**.** *Every configuration induced by application of the instructions produced by the pebbling strategy in Figure 2 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 $3d$ 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 3, 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. □

5 Core Adaptive Security Component

In this section, we will describe the secret-sharing scheme $(\mathsf{share}, \mathsf{reconstruct})$ 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 to define hybrids in the piecewise guessing framework of Section 3.

Overview. As described in the overview in Section 1.1, we will consider the following "core 1-ABE component":

$$\begin{aligned}
&\mathsf{ct}'_{\mathbf{x}} := (\{w_i\}_{x_i=1}) \quad // \text{ "stripped down" } \mathsf{ct}_{\mathbf{x}} \\
&\mathsf{sk}_f := (\{h^{\mu_j}\}_{\rho(j)=0} \cup \{h^{\mu_j + r_j w_{\rho(j)}}, h^{r_j}\}_{\rho(j)\neq 0})
\end{aligned}$$

where $(\{\mu_j\}, \rho) \leftarrow \mathsf{share}(f, \mu)$. We want to show that under the DDH assumption, μ is hidden given just $(\mathsf{ct}'_{\mathbf{x}}, \mathsf{sk}_f)$ where $\mathbf{x}, f$ are adaptively chosen subject to the constraint $f(\mathbf{x}) = 0$. We formalize this via a pair of games $\mathsf{G}_0^{\text{1-ABE}}, \mathsf{G}_1^{\text{1-ABE}}$ and the requirement $\mathsf{G}_0^{\text{1-ABE}} \approx_c \mathsf{G}_1^{\text{1-ABE}}$. In fact, we will study a more abstract construction based on any CPA-secure encryption with:

$$\begin{aligned}
&\mathsf{ct}'_{\mathbf{x}} := (\{w_i\}_{x_i=1}) \quad // \text{ "stripped down" } \mathsf{ct}_{\mathbf{x}} \\
&\mathsf{sk}'_f := \{\mu_j\}_{\rho(j)=0} \cup \{\mathsf{CPA.Enc}(w_{\rho(j)}, \mu_j)\}_{\rho(j)\neq 0} \text{ where } (\{\mu_j\}, \rho) \leftarrow \mathsf{share}(f, \mu)
\end{aligned}$$

5.1 Linear secret sharing for NC^1

We first describe a linear secret-sharing scheme for NC^1; this is essentially the information-theoretic version of Yao's secret-sharing for NC^1 in [20,32,19]. 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, and give an example in Figure 5. 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

$\mathsf{share}(f, \mu)$:
Input: A formula $f : \{0,1\}^n \to \{0,1\}$ of size m and a secret $\mu \in \mathbb{Z}_p$.

1. For each non-output wire $j = 1, ..., m-1$, pick a uniformly random $\hat{\mu}_j \leftarrow \mathbb{Z}_p$. For the output wire, set $\hat{\mu}_m = \mu$
2. For each outgoing wire j from input node i, add $\mu_j = \hat{\mu}_j$ to the output set of shares and set $\rho(j) = i$.
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 $\rho(c) = 0$.
4. For each OR gate g with input wires a, b and output wire c, add $\mu_{c_a} = \hat{\mu}_c + \hat{\mu}_a \in \mathbb{Z}_p$ and $\mu_{c_b} = \hat{\mu}_c + \hat{\mu}_b \in \mathbb{Z}_p$ to the output set of shares and set $\rho(c_a) = 0$ and $\rho(c_b) = 0$.
5. Output $\{\mu_j\}, \rho$.

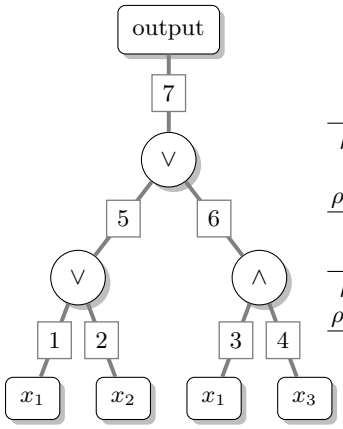
Fig. 4. Information-theoretic linear secret sharing scheme share for NC^1

$|\{\mu_j\}| \leq 2m$ since each of the m nodes adds a single μ_j to the output set, except for OR gates which add two: μ_{j_a} and μ_{j_b}.

The reconstruction procedure $\mathsf{reconstruct}$ 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 μ_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 μ_{c_a} or μ_{c_b} (which are both available since $\rho(c_a) = \rho(c_b) = 0$).

Note that $\mathsf{reconstruct}(f, x, \{\mu_j\}_{\rho(j)=0 \vee x_{\rho(j)}=1})$ computes a linear operation with respect to the shares μ_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, $\mathsf{reconstruct}(f, x, \{\mu_j\}_{\rho(j)=0 \vee x_{\rho(j)}=1})$ is equivalent to identifying the coefficients ω_j of this linear function, where $\mu = \sum_{\rho(j)=0 \vee x_{\rho(j)}=1} \omega_j \mu_j$.



j	1	2	3	4	5_a
μ_j	$\hat{\mu}_1$	$\hat{\mu}_2$	$\hat{\mu}_3$	$\hat{\mu}_4$	$\hat{\mu}_1 + \hat{\mu}_5$
$\rho(j)$	1	2	1	3	0

j	5_b	6	7_a	7_b
μ_j	$\hat{\mu}_2 + \hat{\mu}_5$	$\hat{\mu}_3 + \hat{\mu}_4 + \hat{\mu}_6$	$\hat{\mu}_5 + \mu$	$\hat{\mu}_6 + \mu$
$\rho(j)$	0	0	0	0

Fig. 5. Left: Formula $(x_1 \vee x_2) \vee (x_1 \wedge x_3)$, where the wires are numbered $1, 2, \ldots, 7$. Right: Shares $(\mu_1, \ldots, \mu_{7_b})$ and mapping ρ for the formula corresponding to secret $\mu \in \mathbb{Z}_p$

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

$$\mathsf{share}(f, \mathbf{v}) := (\{\mathbf{v}_j := (v_{1,j}, ..., v_{k,j}))\}, \rho) \text{ where } (\{v_{i,j}\}, \rho) \leftarrow \mathsf{share}(f, v_i)$$

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

$$\mathsf{reconstruct}(f, x, \{\mathbf{v}_j\}_{\rho(j)=0 \vee x_{\rho(j)}=1}) := \sum_{\rho(j)=0 \vee x_{\rho(j)=1}} \omega_j \mathbf{v}_j \text{ where } \omega_j \text{ are computed as above}$$

Our final ABE construction will use this extension.

5.2 Core 1-ABE Security Game

Definition 4 (core 1-ABE security $\mathsf{G}_0^{\mathbf{1\text{-}abe}}, \mathsf{G}_1^{\mathbf{1\text{-}abe}}$). *For a stateful adversary $\mathcal{A}$, we define the following games $\mathsf{G}_\beta^{1\text{-ABE}}$ for $\beta \in \{0, 1\}$.*

$$\langle \mathcal{A}, \mathsf{G}_\beta^{1\text{-ABE}} \rangle := \mathbb{I} \left\{ \begin{array}{l} \mu^{(0)}, \mu^{(1)} \leftarrow \mathbb{Z}_p; w_i \leftarrow \mathsf{CPA.Setup}(\lambda) \\ b' \leftarrow \mathcal{A}^{\mathcal{O}_\mathsf{F}(\cdot), \mathcal{O}_\mathsf{X}(\cdot), \mathcal{O}_\mathsf{E}(\cdot,\cdot)}(\mu^{(0)}) \end{array} \right\}$$

where the adversary $\mathcal{A}$ adaptively interacts with three oracles:

$$\mathcal{O}_\mathsf{F}(f) := \{\mathsf{sk}'_f = \{\mu_j\}_{\rho(j)=0} \cup \{\mathsf{CPA.Enc}(w_{\rho(j)}, \mu_j)\}_{\rho(j)\neq 0} \text{ where } (\{\mu_j\}, \rho) \leftarrow \mathsf{share}(f, \mu^{(\beta)})$$
$$\mathcal{O}_\mathsf{X}(x) := (\mathsf{ct}'_x = \{w_i\}_{x_i=1})$$
$$\mathcal{O}_\mathsf{E}(i, m) := \mathsf{CPA.Enc}_{w_i}(m)$$

with the restrictions that (i) only one query is made to each of $\mathcal{O}_\mathsf{F}(\cdot)$ and $\mathcal{O}_\mathsf{X}(\cdot)$, and (ii) the queries f and x to $\mathcal{O}_\mathsf{F}(\cdot), \mathcal{O}_\mathsf{X}(\cdot)$ respectively, satisfy $f(x) = 0$.

To be clear, the β in $\mathsf{G}_{\beta}^{\text{1-ABE}}$ affects only the implementation of the oracle $\mathcal{O}_{\mathsf{F}}$ (where $\mu^{(\beta)}$ is shared). We will show that $\mathsf{G}_0^{\text{1-ABE}} \approx_c \mathsf{G}_1^{\text{1-ABE}}$ where we instantiate share using the scheme in Section 5.1. That is, Theorem 2 will bound the quantity:

$$\Pr[\langle \mathsf{A}, \mathsf{G}_0^{\text{1-ABE}} \rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{G}_1^{\text{1-ABE}} \rangle = 1]$$

Comparison with [20]. Proving adaptive security for the core 1-ABE with share is very similar to the proof for adaptive secret-sharing for circuits in [20]. 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 [20], f is fixed, and the adaptive choices correspond to x, but revealed one bit at a time (that is, $\mathcal{O}_{\mathsf{X}}(i, x_i)$ returns w_i if $x_i = 1$). Another difference is the $\mathcal{O}_{\mathsf{E}}$ 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 3 and 4 to achieve polynomial security loss, rather than the quasi-polynomial loss we would get from following their proof more directly.

5.3 Adaptive Security for Core 1-ABE Component

We will show that $\mathsf{G}_0^{\text{1-ABE}} \approx_c \mathsf{G}_1^{\text{1-ABE}}$ as defined in Definition 4 using the piecewise guessing framework. To do this, we need to first define a family of games $\{\mathsf{H}^u\}$ along with functions $h_0, \ldots, h_L$, using the pebbling strategy in Section 4. First, we will describe share^u, which will be used to define H^u.

Defining share^u Recall that Lemma 4 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 NC^1 formula, the shares are generated as in the secret-sharing scheme in Figure 4, except for each pebbled node with output wire c, we replace μ_c with an independent random $\mu_c \leftarrow \mathbb{Z}_p$ (in the case of a pebbled OR gate, we replace both associated μ_{c_a} and μ_{c_b} with independent random $\mu_{c_a}, \mu_{c_b} \leftarrow \mathbb{Z}_p$, i.e: both μ_{c_a}, μ_{c_b} are associated with wire c.). In particular, we get the procedure $\mathsf{share}^u(f, \mu)$ defined in Figure 6.

Hybrid Distribution H^u We now define our hybrid games, and remark that Section 3 used $z \in \{0,1\}^R$ to denote the adaptive choices made by an adversary, and the functions h_ℓ that define our hybrid games will depend on the adaptive choices of both the $f \in \mathsf{NC}^1$ and $x \in \{0,1\}^n$ chosen during the game, so in our application of the piecewise guessing framework of Section 3, 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.

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

1. Compute $(\{\mu'_j\}, \rho) \leftarrow \mathsf{share}(f, \mu)$ as defined in Figure 4
2. For each μ'_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 $\{\mu_j\}, \rho$.

Fig. 6. Pebbling-modified secret sharing scheme share^u

Definition 5 (H^u and h_ℓ). *Let H^u be $\mathsf{G}_0^{1\text{-ABE}}$ with $\boxed{\mathsf{share}^u}(f, \mu^{(0)})$ used in the implementation of oracle $\mathcal{O}_\mathsf{F}(f)$ (replacing $\boxed{\mathsf{share}}(f, \mu^{(0)})$). Let $h_\ell : \mathsf{NC}^1 \times \{0,1\}^n \to \{0,1\}^{R'}$ 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 ℓ instructions from $\mathsf{Pebble}(f, x, g^*)$ of Figure 2.*

Note that the first 0 instructions specify a configuration with no pebbles, so h_0 is a constant function for all f, x. Also, from the inductive proof in Lemma 3, we know that all sequences of instructions from $\mathsf{Pebble}(f, x, g^*)$ when $f(x) = 0$ result in a configuration with a single pebble on the root gate g^*, so h_L is a constant function for all f, x where $f(x) = 0$. Furthermore, note that for all such f, x:

- $\mathsf{H}^{h_0(f,x)}$ is equivalent to $\mathsf{G}_0^{1\text{-ABE}}$ (since $\mathsf{share}^{h_0(f,x)}(f, \mu^{(0)}) = \mathsf{share}(f, \mu^{(0)})$);
- $\mathsf{H}^{h_L(f,x)}$ is equivalent to $\mathsf{G}_1^{1\text{-ABE}}$ (since $\mathsf{share}^{h_L(f,x)}(f, \mu^{(0)}) = \mathsf{share}(f, \mu^{(1)})$ 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: μ_m).

We now have a series of hybrids $\mathsf{G}_0^{1\text{-ABE}} \equiv \mathsf{H}^{h_0(f,x)}, \mathsf{H}^{h_1(f,x)}, ..., \mathsf{H}^{h_L(f,x)} \equiv \mathsf{G}_1^{1\text{-ABE}}$ which satisfy end-point equivalence and, according to the piecewise guessing framework described in Section 3, define games $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1), \widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)$ for $\ell \in [0, L]$.

Lemma 5 (neighboring indistinguishability). *For all $\ell \in [L]$ and $u_0, u_1 \in \{0,1\}^{R'}$, $\Pr[\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)\rangle = 1] - \Pr[\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)\rangle = 1] \leq n \cdot \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{B}}(\lambda)$*

Proof. First, observe that the difference between $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)$ and $\widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)$ lies in $\mathcal{O}_\mathsf{F}(\cdot)$: the former uses $\boxed{\mathsf{share}^{u_0}}$ and the latter uses $\boxed{\mathsf{share}^{u_1}}$. Now, fix the adaptive query f to $\mathcal{O}_\mathsf{F}$. 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 $\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)\rangle$ and $\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)\rangle$ 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 $\mathsf{Pebble}(f, x', g^*)$, so they differ by a pebbling instruction that follows one of the rules in Definition 3. 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 $\boxed{j}$: Here, the only difference from $\mathsf{share}^{u_0}(f, \mu^{(0)})$ to $\mathsf{share}^{u_1}(f, \mu^{(0)})$ is that we change $\boxed{\mu_j}$ to a random element of $\mathbb{Z}_p$ (or vice-versa). The pebbling rule for an input node requires that the input x to $\mathcal{O}_{\mathsf{X}}(\cdot)$ in both $\widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)$ and $\widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)$ satisfies $x_{\rho(j)} = 0$. Indistinguishability then follows from the CPA security of $(\mathsf{CPA.Setup}, \mathsf{CPA.Enc}, \mathsf{CPA.Dec})$ 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 $\mathcal{O}_{\mathsf{X}}(x)$. 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 $\mathcal{O}_{\mathsf{X}}(x)$).
- pebble/unpebble AND gate with out-going wire $\boxed{c}$ and input wires a, b corresponding to nodes g_a, g_b. Here, the only difference from $\mathsf{share}^{u_0}(f, \mu^{(0)})$ to $\mathsf{share}^{u_1}(f, \mu^{(0)})$ is that we change $\boxed{\mu_c}$ 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, μ_a is independent and uniformly random in both distributions $\mathsf{share}^{u_0}(f, \mu^{(0)})$ and $\mathsf{share}^{u_1}(f, \mu^{(0)})$, 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 $\mathsf{share}^{u_0}(f, \mu^{(0)})$ and $\mathsf{share}^{u_1}(f, \mu^{(0)})$ are identically distributed, and therefore so is the output $\mathcal{O}_{\mathsf{F}}(f)$. (This holds even if the adversary receives all of $\{w_i : i \in [n]\}$ from its query to $\mathcal{O}_{\mathsf{X}}(x)$).
- pebble/unpebble OR gate with out-going wire $\boxed{c}$ and input wires a, b corresponding to nodes g_a, g_b. Here, the only difference from $\mathsf{share}^{u_0}(f, \mu^{(0)})$ to $\mathsf{share}^{u_1}(f, \mu^{(0)})$ is that we change $\boxed{\mu_{c_a}, \mu_{c_b}}$ 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, μ_a and μ_b are independent and uniformly random in both distributions $\mathsf{share}^{u_0}(f, \mu^{(0)})$ and $\mathsf{share}^{u_1}(f, \mu^{(0)})$, 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 $\mathsf{share}^{u_0}(f, \mu^{(0)})$ and $\mathsf{share}^{u_1}(f, \mu^{(0)})$ are identically distributed, and therefore so is the output $\mathcal{O}_{\mathsf{F}}(f)$. (This holds even if the adversary receives all of $\{w_i : i \in [n]\}$ in its query to $\mathcal{O}_{\mathsf{X}}(x)$).

In all cases, the simulator can return an appropriately distributed answer to $\mathcal{O}_{\mathsf{X}}(x) = \{w_i\}_{x_i=1}$ 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 $\mathcal{O}_{\mathsf{E}}(i, m) = \mathsf{Enc}_{w_i}(m)$, 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 $\mathsf{Enc}_{w_i}(\cdot)$ 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 $\mathcal{O}_{\mathsf{E}}$ requests to that variable with the CPA $\mathsf{Enc}_{w_i}(\cdot)$ 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 $\mathcal{O}_{\mathsf{E}}$ made before $\mathcal{O}_{\mathsf{X}}$, so the simulator must guess uniformly and take a security loss of n).

In all but the input node case, the two distributions $\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)\rangle$ and $\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)\rangle$ 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 $\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{B}}(\lambda)$, conditioned on a correct guess of which of the n input variables corresponds to the pebbled/unpebbled input node. Therefore, $\Pr[\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)\rangle = 1] - \Pr[\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)\rangle = 1] \leq n \cdot \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{B}}(\lambda)$ □

5.4 CPA-secure symmetric encryption

We will instantiate $(\mathsf{CPA.Setup}, \mathsf{CPA.Enc}, \mathsf{CPA.Dec})$ in our Core 1-ABE of Definition 4 with a variant of the standard CPA-secure symmetric encryption scheme based on k-Lin from [11] that supports messages $[M]_2 \in G_2$ of an asymmetric prime-order bilinear group $\mathbb{G}$:

$\mathsf{CPA.Setup}(1^\lambda)$: Run $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$. Sample $\mathbf{M}_0 \leftarrow \mathbb{Z}_p^{k\times k}$, $\mathbf{m}_1 \leftarrow \mathbb{Z}_p^k$,
output $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1) := (\mathbf{M}_0, \mathbf{m}_1^\top)$
$\mathsf{CPA.Enc}(\mathsf{sk}, [M]_2)$: Sample $\mathbf{r} \leftarrow \mathbb{Z}_p^k$, output $(\mathsf{ct}_0, \mathsf{ct}_1) := ([M + \mathbf{m}_1^\top \mathbf{r}]_2, [\mathbf{M}_0 \mathbf{r}]_2)$
$\mathsf{CPA.Dec}((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_0, \mathsf{ct}_1))$: Output $\mathsf{ct}_0 \cdot \mathsf{sk}_1 \cdot \mathsf{sk}_0^{-1} \cdot \mathsf{ct}_1$.

Correctness Note that: $\mathsf{ct}_0 \cdot \mathsf{sk}_1 \cdot \mathsf{sk}_0^{-1} \cdot \mathsf{ct}_1 = [M + \mathbf{m}_1^\top \mathbf{r} - \mathbf{m}_1^\top \mathbf{r}]_2 = [M]_2$.

Lemma 6. $\mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{B}}(\lambda) \leq \mathsf{Adv}^{k\text{-Lin}}_{\mathcal{B}'}(\lambda)$

Proof. Proof is contained in the full version of this paper [22] and omitted here for brevity.

Theorem 2. *The Core 1-ABE component of Definition 4 implemented with* $(\mathsf{share}, \mathsf{reconstruct})$ *from Section 5.1 and the CPA-secure symmetric encryption scheme* $(\mathsf{CPA.Setup}, \mathsf{CPA.Enc}, \mathsf{CPA.Dec})$ *from Section 5.4 satisfies:*

$$\Pr[\langle \mathsf{A}, \mathsf{G}_0^{\text{1-ABE}}\rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{G}_1^{\text{1-ABE}}\rangle = 1] \leq 2^{6d} \cdot 8^d \cdot n \cdot \mathsf{Adv}^{k\text{-Lin}}_{\mathcal{B}^*}(\lambda)$$

Proof. Recall the hybrids $\mathsf{G}_0^{\text{1-ABE}} \equiv \mathsf{H}^{h_0(f,x)}, \mathsf{H}^{h_1(f,x)}, ..., \mathsf{H}^{h_L(f,x)} \equiv \mathsf{G}_1^{\text{1-ABE}}$ defined in Section 5.3. Lemma 5 tells us that: for all $\ell \in [L]$ and $u_0, u_1 \in \{0,1\}^{R'}$,

$$\Pr[\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,0}(u_0, u_1)\rangle = 1] - \Pr[\langle \mathsf{A}, \widehat{\mathsf{H}}_{\ell,1}(u_0, u_1)\rangle = 1] \leq n \cdot \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{B}}(\lambda)$$

These hybrids satisfy the end-point equivalence requirement, so Lemma 1 then tells us that:

$$\Pr[\langle \mathsf{A}, \mathsf{G}_0^{\text{1-ABE}}\rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{G}_1^{\text{1-ABE}}\rangle = 1] \leq 2^{2R'} \cdot L \cdot n \cdot \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{B}}(\lambda)$$

Lemma 4 tells us that $R' \leq 3d$, and Lemma 2 tells us that $L \leq 8^d$, where d is the depth of the formula. Finally, Lemma 6 tells us that $\mathsf{Adv}^{\text{CPA}}_{\mathcal{B}}(\lambda) \leq \mathsf{Adv}^{k\text{-Lin}}_{\mathcal{B}^*}(\lambda)$. So: $\Pr[\langle \mathsf{A}, \mathsf{G}_0^{\text{1-ABE}} \rangle = 1] - \Pr[\langle \mathsf{A}, \mathsf{G}_1^{\text{1-ABE}} \rangle = 1] \leq 2^{6d} \cdot 8^d \cdot n \cdot \mathsf{Adv}^{k\text{-Lin}}_{\mathcal{B}^*}(\lambda)$ □

6 Our KP-ABE Scheme

In this section, we present our compact KP-ABE for NC^1 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 1.1, we incorporated optimizations from [15,5] to shrink $\mathbf{W}_i$ and thus the secret key, and hence the need for the $\mathcal{O}_{\mathsf{E}}$ oracle in the core 1-ABE security game.

6.1 The scheme

Our KP-ABE scheme is as follows:

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

$$\mathbf{A} \leftarrow \mathbb{Z}_p^{k \times (k+1)}, \mathbf{W}_i \leftarrow \mathbb{Z}_p^{(k+1) \times k} \; \forall i \in [n], \mathbf{v} \leftarrow \mathbb{Z}_p^{k+1}$$

and output:

$$\begin{aligned} \mathsf{msk} &:= (\ \mathbf{v}, \mathbf{W}_1, \ldots, \mathbf{W}_n\ ) \\ \mathsf{mpk} &:= (\ [\mathbf{A}]_1, [\mathbf{A}\mathbf{W}_1]_1, \ldots, [\mathbf{A}\mathbf{W}_n]_1,\ e([\mathbf{A}]_1, [\mathbf{v}]_2)\ ) \end{aligned}$$

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

$$\begin{aligned} \mathsf{ct}_{\mathbf{x}} &= (\mathsf{ct}_1, \{\mathsf{ct}_{2,i}\}_{x_i=1}, \mathsf{ct}_3) \\ &:= \left([\mathbf{s}^\top \mathbf{A}]_1, \{[\mathbf{s}^\top \mathbf{A}\mathbf{W}_i]_1\}_{x_i=1}, \quad e([\mathbf{s}^\top \mathbf{A}]_1, [\mathbf{v}]_2) \cdot M \right) \end{aligned}$$

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, f)$: Sample $(\{\mathbf{v}_j\}, \rho) \leftarrow \mathsf{share}(f, \mathbf{v})$, $\mathbf{r}_j \leftarrow \mathbb{Z}_p^k$. Output:

$$\begin{aligned} \mathsf{sk}_f &= (\{\mathsf{sk}_{1,j}, \mathsf{sk}_{2,j}\}) \\ &:= (\ \{[\mathbf{v}_j + \mathbf{W}_{\rho(j)}\mathbf{r}_j]_2, [\mathbf{r}_j]_2\}\ ) \end{aligned}$$

where $\mathbf{W}_0 = \mathbf{0}$.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_f, \mathsf{ct}_{\mathbf{x}})$: Compute ω_j such that $\mathbf{v} = \sum_{\rho(j)=0 \vee x_{\rho(j)}=1} \omega_j \mathbf{v}_j$ as described in Section 5.1. Output:

$$\mathsf{ct}_3 \cdot \prod_{\rho(j)=0 \vee x_{\rho(j)}=1} \left(\frac{e(\mathsf{ct}_{2,\rho(j)}, \mathsf{sk}_{2,j})}{e(\mathsf{ct}_1, \mathsf{sk}_{1,j})} \right)^{\omega_j}$$

6.2 Correctness

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

$$\frac{e(\mathsf{ct}_1, \mathsf{sk}_{1,j})}{e(\mathsf{ct}_{2,\rho(j)}, \mathsf{sk}_{2,j})} = [\mathbf{s}^\top \mathbf{A} \mathbf{v}_j]_T$$

which follows from the fact that

$$\mathbf{s}^\top \mathbf{A} \mathbf{v}_j = \underbrace{\mathbf{s}^\top \mathbf{A}}_{\mathsf{ct}_1} \cdot \underbrace{(\mathbf{v}_j + \mathbf{W}_{\rho(j)} \mathbf{r}_j)}_{\mathsf{sk}_{1,j}} - \underbrace{\mathbf{s}^\top \mathbf{A} \mathbf{W}_{\rho(j)}}_{\mathsf{ct}_{2,\rho(j)}} \cdot \underbrace{\mathbf{r}_j}_{\mathsf{sk}_{2,j}}$$

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

$$\begin{aligned}
\mathsf{ct}_3 \cdot \prod_{\rho(j)=0 \vee x_{\rho(j)}=1} \left(\frac{e(\mathsf{ct}_{2,\rho(j)}, \mathsf{sk}_{2,j})}{e(\mathsf{ct}_1, \mathsf{sk}_{1,j})} \right)^{\omega_j} &= M \cdot [\mathbf{s}^\top \mathbf{A} \mathbf{v}]_T \cdot \prod_{\rho(j)=0 \vee x_{\rho(j)}=1} [\mathbf{s}^\top \mathbf{A} \mathbf{v}_j]_T^{-\omega_j} \\
&= M \cdot [\mathbf{s}^\top \mathbf{A} \mathbf{v}]_T \cdot [-\mathbf{s}^\top \mathbf{A} \sum_{\rho(j)=0 \vee x_{\rho(j)}=1} \omega_j \mathbf{v}_j]_T \\
&= M \cdot [\mathbf{s}^\top \mathbf{A} \mathbf{v}]_T \cdot [-\mathbf{s}^\top \mathbf{A} \mathbf{v}]_T \\
&= M
\end{aligned}$$

6.3 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 $\mathbf{s}^\top \mathbf{A}$ replaced with $\mathbf{c}^\top \leftarrow \mathbb{Z}_p^{k+1}$. That is, $\mathsf{ct}_{\mathbf{x}} := \left([\boxed{\mathbf{c}^\top}]_1, \{[\boxed{\mathbf{c}^\top} \mathbf{W}_i]_1\}_{x_i=1}, \quad e([\boxed{\mathbf{c}^\top}]_1, [\mathbf{v}]_2) \cdot M \right)$

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

- Normal: generated as in the scheme.
- SF: same as a Normal key, except $\mathbf{v}$ replaced with $\mathbf{v} + \delta \mathbf{a}^\perp$, where a fresh $\delta \leftarrow \mathbb{Z}_p$ is chosen per SF key and $\mathbf{a}^\perp$ is any fixed $\mathbf{a}^\perp \in \mathbb{Z}_p^{k+1} \setminus \{\mathbf{0}\}$ such that $\mathbf{A}\mathbf{a}^\perp = \mathbf{0}$. That is, $\mathsf{sk}_f := (\{[\mathbf{v}_j + \mathbf{W}_{\rho(j)} \mathbf{r}_j]_2, [\mathbf{r}_j]_2\})$ where $(\{\mathbf{v}_j\}, \rho) \leftarrow \mathsf{share}(f, \boxed{\mathbf{v} + \delta \mathbf{a}^\perp})$, $\mathbf{r}_j \leftarrow \mathbb{Z}_p^k$.

SF stands for semi-functional following the terminology in previous works [25,33].

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.
- $\mathsf{H}_{2,\ell}, \ell = 0, \ldots, Q$: same as H_1, except the first ℓ keys are SF and the remaining $Q - \ell$ keys are Normal.
- H_3: replace M with random $\widetilde{M}$.

Proof overview.

- We have $\mathsf{H}_0 \approx_c \mathsf{H}_1 \equiv \mathsf{H}_{2,0}$ via k-Lin, which tells us $([\mathbf{A}]_1, [\mathbf{s}^\top \mathbf{A}]_1) \approx_c ([\mathbf{A}]_1, [\mathbf{c}^\top]_1)$. Here, the security reduction will pick $\mathbf{W}_1, \ldots, \mathbf{W}_n$ and $\mathbf{v}$ so that it can simulate the mpk, the ciphertext and the secret keys.
- We have $\mathsf{H}_{2,\ell-1} \approx_c \mathsf{H}_{2,\ell}$, for all $\ell \in [Q]$. The difference between the two is that we switch the ℓ'th sk_f from Normal to SF using the adaptive security of our core 1-ABE component in $\mathsf{G}^{\text{1-ABE}}$ from Section 5. The idea is to sample

$$\mathbf{v} = \tilde{\mathbf{v}} + \mu \mathbf{a}^\perp, \mathbf{W}_i = \widetilde{\mathbf{W}}_i + \mathbf{a}^\perp \mathbf{w}_i^\top$$

  so that mpk can be computed using $\tilde{\mathbf{v}}, \widetilde{\mathbf{W}}_i$ and perfectly hide $\mu, \mathbf{w}_1, \ldots, \mathbf{w}_n$. Roughly speaking: the reduction
  - uses $\mathcal{O}_{\mathsf{X}}(x)$ in $\mathsf{G}^{\text{1-ABE}}$ to simulate the challenge ciphertext
  - uses $\mathcal{O}_{\mathsf{F}}(f)$ in $\mathsf{G}^{\text{1-ABE}}$ to simulate ℓ'th secret key
  - uses $\mu^{(0)}$ from $\mathsf{G}^{\text{1-ABE}}$ together with $\mathcal{O}_{\mathsf{E}}(i, \cdot) = \mathsf{Enc}(w_i, \cdot)$ to simulate the remaining $Q - \ell$ secret keys
- We have $\mathsf{H}_{2,Q} \equiv \mathsf{H}_3$. In $\mathsf{H}_{2,Q}$, the secret keys only leak $\mathbf{v} + \delta_1 \mathbf{a}^\perp, \ldots, \mathbf{v} + \delta_Q \mathbf{a}^\perp$. This means that $\mathbf{c}^\top \mathbf{v}$ is statistically random (as long as $\mathbf{c}^\top \mathbf{a}^\perp \neq 0$).

Theorem 3 (adaptive KP-ABE). *The KP-ABE construction in Section 6.1 is adaptively secure under the MDDH$_k$ assumption.*

Proof. The detailed proof is contained in the full version of this paper [22] and omitted here for brevity.

Acknowledgments. We thank Allison Bishop, Sanjam Garg, Rocco Servedio, and Daniel Wichs for helpful discussions.

References

1. S. Agrawal and M. Chase. Simplifying design and analysis of complex predicate encryption schemes. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 627–656. Springer, Heidelberg, Apr. / May 2017.
2. N. Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 557–577. Springer, Heidelberg, May 2014.
3. N. Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 591–623. Springer, Heidelberg, Dec. 2016.
4. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Press, May 2007.
5. O. Blazy, E. Kiltz, and J. Pan. (Hierarchical) identity-based encryption from affine message authentication. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, Heidelberg, Aug. 2014.

6. J. Chen, R. Gay, and H. Wee. Improved dual system ABE in prime-order groups via predicate encodings. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, Heidelberg, Apr. 2015.
7. J. Chen, J. Gong, L. Kowalczyk, and H. Wee. Unbounded ABE via bilinear entropy expansion, revisited. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 503–534. Springer, Heidelberg, Apr. / May 2018.
8. J. Chen and H. Wee. Fully, (almost) tightly secure IBE and dual system groups. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 435–460. Springer, Heidelberg, Aug. 2013.
9. J. Chen and H. Wee. Semi-adaptive attribute-based encryption and improved delegation for Boolean formula. In M. Abdalla and R. D. Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 277–297. Springer, Heidelberg, Sept. 2014.
10. J. H. Cheon. Security analysis of the strong Diffie-Hellman problem. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–11. Springer, Heidelberg, May / June 2006.
11. A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, Aug. 2013.
12. G. Fuchsbauer, Z. Jafargholi, and K. Pietrzak. A quasipolynomial reduction for generalized selective decryption on trees. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 601–620. Springer, Heidelberg, Aug. 2015.
13. G. Fuchsbauer, M. Konstantinov, K. Pietrzak, and V. Rao. Adaptive security of constrained PRFs. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, Heidelberg, Dec. 2014.
14. S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 479–499. Springer, Heidelberg, Aug. 2013.
15. J. Gong, X. Dong, J. Chen, and Z. Cao. Efficient IBE with tight reduction to standard assumption in the multi-challenge setting. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 624–654. Springer, Heidelberg, Dec. 2016.
16. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013.
17. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 89–98. ACM Press, Oct. / Nov. 2006. Available as Cryptology ePrint Archive Report 2006/309.
18. B. Hemenway, Z. Jafargholi, R. Ostrovsky, A. Scafuro, and D. Wichs. Adaptively secure garbled circuits from one-way functions. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 149–178. Springer, Heidelberg, Aug. 2016.
19. Y. Ishai and E. Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In P. Widmayer, F. T. Ruiz, R. M. Bueno, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *ICALP 2002*, volume 2380 of *LNCS*, pages 244–256. Springer, Heidelberg, July 2002.

20. Z. Jafargholi, C. Kamath, K. Klein, I. Komargodski, K. Pietrzak, and D. Wichs. Be adaptive, avoid overcommitting. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 133–163. Springer, Heidelberg, Aug. 2017.
21. Z. Jafargholi and D. Wichs. Adaptive security of Yao's garbled circuits. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 433–458. Springer, Heidelberg, Oct. / Nov. 2016.
22. L. Kowalczyk and H. Wee. Compact adaptively secure ABE for NC1 from k-Lin. *IACR Cryptology ePrint Archive*, 2019:224, 2019.
23. A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May / June 2010.
24. A. B. Lewko, A. Sahai, and B. Waters. Revocation systems with very small private keys. In *2010 IEEE Symposium on Security and Privacy*, pages 273–285. IEEE Computer Society Press, May 2010.
25. A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479. Springer, Heidelberg, Feb. 2010.
26. A. B. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 180–198. Springer, Heidelberg, Aug. 2012.
27. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, Aug. 2010.
28. T. Okamoto and K. Takashima. Fully secure unbounded inner-product and attribute-based encryption. In X. Wang and K. Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 349–366. Springer, Heidelberg, Dec. 2012.
29. R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM CCS 07*, pages 195–203. ACM Press, Oct. 2007.
30. B. Parno, M. Raykova, and V. Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 422–439. Springer, Heidelberg, Mar. 2012.
31. A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.
32. V. Vinod, A. Narayanan, K. Srinathan, C. P. Rangan, and K. Kim. On the power of computational secret sharing. In T. Johansson and S. Maitra, editors, *INDOCRYPT 2003*, volume 2904 of *LNCS*, pages 162–176. Springer, Heidelberg, Dec. 2003.
33. B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, Aug. 2009.
34. H. Wee. Dual system encryption via predicate encodings. In Y. Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 616–637. Springer, Heidelberg, Feb. 2014.