Compact Adaptively Secure ABE from k-Lin: Beyond NC^1 and towards NL

Huijia Lin and Ji Luo

University of Washington
{rachel,luoji}@cs.washington.edu

Abstract. We present a new general framework for constructing *compact* and *adaptively secure* attribute-based encryption (ABE) schemes from k-Lin in asymmetric bilinear pairing groups. Previously, the only construction [Kowalczyk and Wee, Eurocrypt '19] that simultaneously achieves compactness and adaptive security from static assumptions supports policies represented by *Boolean formulae*. Our framework enables supporting more expressive policies represented by *arithmetic branching programs*.

Our framework extends to ABE for policies represented by uniform models of computation such as Turing machines. Such policies enjoy the feature of being applicable to attributes of arbitrary lengths. We obtain the first compact adaptively secure ABE for deterministic and non-deterministic finite automata (DFA and NFA) from k-Lin, previously unknown from any static assumptions. Beyond finite automata, we obtain the first ABE for large classes of uniform computation, captured by deterministic and non-deterministic *logspace* Turing machines (the complexity classes L and NL) based on k-Lin. Our ABE scheme has compact secret keys of size linear in the description size of the Turing machine M. The ciphertext size grows linearly in the input length, but also linearly in the time complexity, and exponentially in the space complexity. Irrespective of compactness, we stress that our scheme is the first that supports large classes of Turing machines based solely on standard assumptions. In comparison, previous ABE for general Turing machines all rely on strong primitives related to indistinguishability obfuscation.

1 Introduction

Attribute-based encryption (ABE) [32] is an advanced form of public-key encryption that enables fine-grained access control. The encryption algorithm using the master public key mpk can encrypt a message m with a descriptive attribute x,[1] producing a ciphertext $\mathsf{ct}_x(m)$. The key generation algorithm using the master secret key msk can produce a secret key sk_y associated with an access policy y. Decrypting $\mathsf{ct}_x(m)$ using sk_y reveals the message m if the attribute x satisfies the policy y; otherwise, no information about m is revealed. The security

[1] Some works call x a set of attributes, and each bit or component of x an attribute. We treat the attribute as a single vector.

requirement of ABE stipulates resilience to collusion attacks — any group of users holding secret keys for different policies learn nothing about the plaintext as long as none of them is individually authorized to decrypt the ciphertext.

A primary goal of research on ABE is designing ABE schemes for expressive classes of policies, usually defined by computation models or complexity classes. A beautiful and fruitful line of works have constructed ABE for many different policy classes. For non-uniform computation, we have ABE for Boolean [32,44] or arithmetic formulae, branching/span programs [13,23,31,37,45,47,48,53,54], and circuits [14,19,30]. For uniform computation, we have ABE for deterministic finite automata [1,5,12,13,29,58], non-deterministic finite automata [4], and even Turing machines [3,8]. These constructions, however, achieve different trade-offs between security, efficiency, and underlying computational assumptions. It is rare to have a construction that simultaneously achieves the following natural desirata on all fronts:

- *Security:* (full) adaptive security (as opposed to selective or semi-adaptive security);
- *Efficiency:* having compact secret key and ciphertext, whose sizes grow linearly with the description size of the policy and the length of the attribute, respectively;
- *Assumptions:* relying on standard and simple assumptions, such as LWE and k-Lin or SXDH in bilinear pairing groups (in particular, it is preferable to avoid the use of strong primitives such as indistinguishability obfuscation, and instance-dependent assumptions such as q-type assumptions, whose strength can be weakened by adversarially chosen parameters).

All previous constructions of ABE fail to achieve at least one of the desirable properties, except for the recent construction of ABE for *Boolean formulae* from the k-Lin assumption by Kowalczyk and Wee [44]. This raises the question:

Can we construct ABE schemes with all the desirable properties above for more expressive classes of policies than Boolean formulae?

When it comes to uniform computation, the state of affairs is even less satisfactory. All constructions of ABE for general Turing machines are based on strong primitives such as indistinguishability obfuscation and multilinear map. Without these powerful tools, existing schemes can only handle the weak computation model of finite automata.

Can we construct ABE schemes based on standard assumptions for more expressive uniform computations than finite automata?

Our Result. Via a unified framework, we construct *compact and adaptively secure* ABE schemes based on the k-Lin assumption in asymmetric prime-order bilinear pairing groups for the following classes of policies:

Arithmetic Branching Programs. ABPs capture many functions of interest, including arithmetic computations like sparse polynomials, mean, and variance, as well as combinatorial computations like string-matching, finite automata, and

decision trees. It is also known that Boolean/arithmetic formulae and Boolean branching programs can all be converted into ABPs with polynomial blow-up in description size. Thus, ABPs can be viewed as a more powerful computational model than them.

Previous ABE schemes for ABPs only provide selective security [30,37] or do not have compact ciphertexts [23].[2] In addition to achieving both adaptive security and compactness, our scheme is the first one that handles ABPs directly without converting it to circuits or arithmetic span programs, which leads to an efficiency improvement in the size of the secret keys from up to quadratic to linear in the size of the ABP.[3]

(Non-)Deterministic Logspace Turing Machines (L *and* NL*).* Here, a secret key is associated with a Turing machine M, and the attribute in a ciphertext specifies an input $\mathbf{x}$, a polynomial time bound T, and a logarithmic space bound S. Decryption succeeds if and only if M accepts $\mathbf{x}$ within time T and space S. Our scheme is *unbounded* in the sense that the public parameters do not restrict the sizes of the Turing machine M and input $\mathbf{x}$, nor the time/space bounds T, S. Furthermore, it enjoys the advantage of ABE for uniform computation that a secret key for M can decrypt ciphertexts with arbitrarily long inputs and arbitrary time/space bounds. This stands in contrast with ABE for non-uniform computation (like ABPs), where a program or circuit f takes inputs of a specific length n, and a secret key for f decrypts only ciphertext of length-n inputs. Achieving this feature is precisely the challenge in constructing ABE for uniform models of computation.

Our scheme is the first ABE for large classes of Turing machine computation, captured by the complexity classes L and NL, *without using the heavy machineries* of multilinear map, extractable witness encryption, or indistinguishability obfuscation as in previous works [3,9,27,41]. In addition, our scheme is adaptively secure and *half-compact*. The secret keys are compact, of size $O(|M|)$ linear in the description size of M, while the ciphertext size depends linearly in $|\mathbf{x}|TS2^S$ (both ignoring fixed polynomial factors in the security parameter).

Removing the dependency on 2^S or T is an interesting open problem that requires technical breakthrough. In particular, removing the dependency on 2^S would give an ABE for polynomial-time Turing machine computation from pairing, a long sought-after goal that has remained elusive for more than a decade. Removing the dependency of encryption time on T even only in the 1-key 1-ciphertext setting implies a succinct message-hiding encoding [42],[4] which is

[2] More precisely, they construct ABE for read-once branching programs. For general branching programs, one can duplicate each component in the attribute for the number of times it is accessed [43]. As such, the ciphertext size grows linearly with the size of the branching program.

[3] An ABP is specified by a directed graph, with edges weighted by affine functions of the input. The size of an ABP is measured by the number of vertices (instead of edges) in the graph.

[4] Message-hiding encodings [42] are a weaker variant of randomized encodings that allow encoding a public computation f, x with a secret message m such that the

only known from strong primitives like indistinguishability obfuscation or functional encryption [18,21,41,42]. Removing the dependency of ciphertext size on T might be an easier task, but would need new techniques different from ours.

Finite Automata. As a special case of ABE for L and NL, we obtain ABE for deterministic finite automata (DFA) and non-deterministic finite automata (NFA).[5] This simply follows from the fact that DFA and NFA can be represented as simple deterministic and non-deterministic Turing machines with space complexity 1 and time complexity N that always move the input tape pointer to the right and never use the work tape.

Previous schemes for DFA based on pairing either achieve only selective security [5,29,58] or rely on q-type assumptions [1,12,13]. The only direct construction of ABE for NFA [4] based on LWE, however, is symmetric-key and only selectively secure. We settle the open problem of constructing adaptively secure ABE for DFA from static assumptions [29] and that of constructing ABE for NFA that is public-key, adaptively secure, or based on assumptions other than LWE [4].

New Techniques for Constructing Adaptively Secure ABE. Constructing adaptively secure ABE is a challenging task. Roughly speaking, previous constructions proceed in two steps. First, a secure core secret-key ABE component for a single ciphertext and a single secret key — termed 1-ABE — is designed. Then, Dual System Encryption framework, originally proposed in [57] and refined in [1,12,13,22,59], provides guidance on how to lift 1-ABE to the public-key and multi-secret-key setting. The main technical challenge lies in the first step: Adaptively secure schemes prior to that of Kowalczyk and Wee [44] either impose a read-once restriction on the attribute[6] [45,59] or rely on q-type assumptions [1,12,16,48]. Kowalczyk and Wee [44] elegantly applied the "partial selectivization" framework [6,38] for achieving adaptive security in general to constructing 1-ABE. In particular, they used a variant of the secret-sharing scheme for Boolean formulae in [38] whose selective simulation security can be proven via a sequence of hybrids, each only requiring *partial* information of the input to be chosen selectively. Then, to show adaptive security, the reduction can *guess* this partial information while incurring only a polynomial security loss.

However, secret-sharing schemes as needed in [44] are only known for Boolean formulae. When dealing with computation over arithmetic domains of potentially exponential size, we have the additional challenge that it is hard to guess even

---

encoding reveals m if and only if $f(x) = 1$. Such encodings are *succinct* if the time to encode is much smaller than the running time of the computation. A pair of ABE secret key for predicate f and ciphertext for attribute x and message m is a message-hiding encoding.

[5] DFA and NFA both characterize regular languages, yet a DFA recognizing a language could have exponentially more states than an NFA recognizing the same language. In this work, by ABE for DFA/NFA, we mean ABE schemes that run in time polynomial in the description size of the finite automata.

[6] As mentioned in Footnote 2, read-once restriction can be circumvented by duplicating attribute components at the cost of losing ciphertext compactness.

a single component of the input, except with exponentially small probability, rendering the partial selectivization framework ineffective. When dealing with uniform computation, we further encounter the challenge that neither the secret key nor the ciphertext is as large as the secret-sharing, making it impossible to directly use information-theoretically secure secret-sharing schemes. We develop new techniques to overcome these challenges.

1. First, we present a generic framework for constructing adaptively secure 1-ABE from *i*) an information theoretic primitive called *arithmetic key garbling*, and *ii*) a computational primitive called *function-hiding inner-product functional encryption* (IPFE) [17,40,49,56]. Our arithmetic key garbling schemes are partial garbling schemes [37] with special structures, which act as the counterpart of secret-sharing schemes for arithmetic computation. Our framework is modular: It decomposes the task of constructing 1-ABE to *first* designing an arithmetic key garbling scheme for the computation class of interest, and *second* applying a generic transformation depending solely on structural properties of the garbling and agnostic of the underlying computation. In particular, the security proof of the transformation does not attempt to trace the computation, unlike [29,44].
2. Second, we formulate structural properties of arithmetic key garbling schemes — called piecewise security — sufficient for achieving adaptive security. The properties are natural and satisfied by the garbling scheme for ABPs in [37]. For logspace Turing machine computation, we present a simple arithmetic key garbling scheme for L and NL, inspired by the garbling schemes in [11,19].
3. Third, we present a new method of lifting 1-ABE to full-fledged ABE using function-hiding IPFE. Our method can be cast into the dual system encryption framework, but is natural on its own, without seeing through the lens of dual system encryption. One feature of IPFE is that it provides a conceptually simple abstraction which allows moving information between ABE keys and ciphertexts easily, and hides away lower-level details on how to guarantee security. This feature makes it a convenient tool in many other parts of the security proof as well.
4. Lastly, to overcome the unique challenge related to ABE for uniform computation, we further enhance our generic method to be able to use partial garbling generated with *pseudorandomness* so that the total size of the secret keys and ciphertexts can be smaller than the garbling.

Organization. In Section 2, we give an overview of our framework for constructing compact adaptively secure ABE schemes for ABPs, logspace Turing machines, and finite automata, using as tools IPFE and arithmetic key garbling schemes (AKGS, a refinement of partial garbling schemes). After introducing basic notations and definitions in Section 3, we define AKGS and its security in Section 4. In Section 5, we show how to construct 1-ABE (the core component of our ABE schemes) for ABPs from an AKGS. Due to space constraints, the security proof of our 1-ABE for ABPs, the construction of full-fledged ABE for ABPs, and ABE for L and NL are provided in the full version.

2 Technical Overview

We now give an overview of our technique, starting with introducing the two key tools arithmetic key garbling schemes and IPFE. Below, by bilinear pairing groups, we mean asymmetric prime-order bilinear pairing groups, denoted as $(G_1, G_2, G_T, g_1, g_2, e)$ and implicitly, $g_T = e(g_1, g_2)$. We use $[\![a]\!]_b$ to represent the encoding g_b^a of a in group G_b.

Arithmetic Key Garbling Scheme. We use a refinement of the notion of partial garbling schemes [37] (which in turn is based on the notion of garbling and randomized encoding [10,36,61]). An *arithmetic key garbling scheme* (AKGS) is an information-theoretic partial garbling scheme for computing $\alpha f(\mathbf{x}) + \beta$ that hides the *secrets* $\alpha, \beta \in \mathbb{Z}_p$, but not $f, \mathbf{x}$:

- A garbling procedure $(\mathbf{L}_1, \ldots, \mathbf{L}_m) \leftarrow \mathsf{Garble}(f, \alpha, \beta; \mathbf{r})$ turns f and two secrets α, β (using randomness $\mathbf{r}$) into m affine *label functions* $L_1, \ldots, L_m$, described by their coefficient vectors $\mathbf{L}_1, \ldots, \mathbf{L}_m$ over $\mathbb{Z}_p$. The label functions specify how to encode an input $\mathbf{x}$ to produce the *labels* for computing $f(\mathbf{x})$ with secrets α, β:

$$\widehat{f(\mathbf{x})}_{\alpha,\beta} = (\ell_1, \ldots, \ell_m), \text{ where } \ell_j = L_j(\mathbf{x}) = \langle \mathbf{L}_j, (1, \mathbf{x}) \rangle \text{ over } \mathbb{Z}_p. \quad (1)$$

- A *linear* evaluation procedure $\gamma \leftarrow \mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m)$ recovers the sum $\gamma = \alpha f(\mathbf{x}) + \beta$ weighted by the function value $f(\mathbf{x})$.

AKGS is a *partial* garbling as it only hides information of the secrets α and β beyond the weighted sum $\alpha f(\mathbf{x}) + \beta$, and does not hide $(f, \mathbf{x})$, captured by a simulation procedure $(\ell'_1, \ldots, \ell'_m) \xleftarrow{\$} \mathsf{Sim}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta)$ that produces the same distribution as the honest labels.

Ishai and Wee [37] proposed a partial garbling scheme for ABPs, which directly implies an AKGS for ABPs. It is also easy to observe that the (fully secure) garbling scheme for arithmetic formulae in [11] can be weakened [19] to an AKGS. Later, we will introduce additional structural and security properties of AKGS needed for our 1-ABE construction. These properties are natural and satisfied by both schemes [11,37].

Inner-Product Functional Encryption. A *function-hiding* (secret-key) inner-product functional encryption (IPFE)[7] enables generating many secret keys $\mathsf{isk}(\mathbf{v}_j)$ and ciphertexts $\mathsf{ict}(\mathbf{u}_i)$ associated with vectors $\mathbf{v}_j$ and $\mathbf{u}_i$ such that decryption yields all the inner products $\{\langle \mathbf{u}_i, \mathbf{v}_j \rangle\}_{i,j} \pmod p$ and nothing else. In this work, we need an *adaptively secure* IPFE, whose security holds even against adversaries choosing all the vectors adaptively. Such an IPFE scheme can be constructed based on the k-Lin assumption in bilinear pairing groups [50,60]. The known scheme also has nice structural properties that will be instrumental to our construction of ABE:

[7] Some works use "inner-product encryption" (IPE) to refer to IPFE [17,25,49,50] and some others [24,39,52–55] use it for inner-product *predicate* encryption.

– $\mathsf{isk}(\mathbf{v}) \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}, [\![\mathbf{v}]\!]_2)$ operates *linearly* on $\mathbf{v}$ (in the exponent of G_2) and the size of the secret key $\mathsf{isk}(\mathbf{v})$ grows linearly with $|\mathbf{v}|$.
– $\mathsf{ict}(\mathbf{u}) \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{msk}, [\![\mathbf{u}]\!]_1)$ also operates *linearly* on $\mathbf{u}$ (in the exponent of G_1) and the size of the ciphertext $\mathsf{ict}(\mathbf{u})$ grows linearly with $|\mathbf{u}|$.
– $\mathsf{IPFE.Dec}(\mathsf{sk}(\mathbf{v}), \mathsf{ct}(\mathbf{u}))$ simply invokes pairing to compute the inner product $[\![\langle \mathbf{u}, \mathbf{v} \rangle]\!]_T$ in the exponent of the target group.

2.1 1-ABE from Arithmetic Key Garbling and IPFE Schemes

1-ABE is the technical heart of our ABE construction. It works in the setting where a single ciphertext $\mathsf{ct}(\mathbf{x})$ for an input vector $\mathbf{x}$ and a single secret key $\mathsf{sk}(f, \mu)$ for a policy $y = f_{\neq 0}$ and a secret μ are published. Decryption reveals μ if $f(\mathbf{x}) \neq 0$; otherwise, μ is hidden.[8]

1-ABE. To hide μ conditioned on $f(\mathbf{x}) = 0$, our key idea is using IPFE to compute an AKGS garbling $\widehat{f(\mathbf{x})}_{\mu,0}$ of $f(\mathbf{x})$ with secrets $\alpha = \mu$ and $\beta = 0$. The security of AKGS guarantees that only $\mu f(\mathbf{x})$ is revealed, which information theoretically hides μ when $f(\mathbf{x}) = 0$.

The reason that it is possible to use IPFE to compute the garbling is attributed to the *affine input-encoding* property of AKGS — the labels $\ell_1, \ldots, \ell_m$ are the output of affine functions $L_1, \ldots, L_j$ of $\mathbf{x}$ as described in Equation (1). Since f, α, β are known at key generation time, the ABE key can be a collection of IPFE secret keys, each encoding the coefficient vector $\mathbf{L}_j$ of one label function L_j. On the other hand, the ABE ciphertext can be an IPFE ciphertext encrypting $(1, \mathbf{x})$. When put together for decryption, they reveal exactly the labels $L_1(\mathbf{x}), \ldots, L_m(\mathbf{x})$, as described below on the left.

Honest Algorithms	Hybrid for Selective Security
$\mathsf{ct}(\mathbf{x})$: $\mathsf{ict}(\ (1, \mathbf{x}) \parallel \mathbf{0}\ )$	$\mathsf{ct}(\mathbf{x})$: $\mathsf{ict}(\ (1, \mathbf{x}) \parallel 1 \parallel \mathbf{0}\ )$
$\mathsf{sk}(f, \mu)$: $j \in [m]$: $\mathsf{isk}_j(\ \mathbf{L}_j \parallel \mathbf{0}\ )$	$\mathsf{sk}(f, \mu)$: $j \in [m]$: $\mathsf{isk}_j(\ \mathbf{0} \parallel \ell_j \parallel \mathbf{0}\ )$

We note that the positions or slots at the right end of the vectors encoded in isk and ict are set to zero by the honest algorithms — $\mathbf{0}$ denotes a vector (of unspecified length) of zeros. These slots provide programming space in the security proof.

It is extremely simple to prove selective (or semi-adaptive) security, where the input $\mathbf{x}$ is chosen before seeing the sk. By the function-hiding property of IPFE, it is indistinguishable to switch the secret keys and the ciphertext to encode any vectors that preserve the inner products. This allows us to hardwire honestly generated labels $\widehat{f(\mathbf{x})}_{\mu,0} = \{\ell_j \leftarrow \langle \mathbf{L}_j, (1, \mathbf{x}) \rangle\}_{j \in [m]}$ in the secret keys as described above on the right. The simulation security of AKGS then implies that only $\mu f(\mathbf{x})$ is revealed, i.e., nothing about μ is revealed.

[8] We can also handle policies of the form $f_{=0}$ so that μ is revealed if and only if $f(\mathbf{x}) = 0$. For simplicity, we focus on one case in this overview.

Achieving Adaptive Security. When it comes to adaptive security, where the input $\mathbf{x}$ is chosen *after* seeing sk, we can no longer hardwire the honest labels $\widehat{f(\mathbf{x})}_{\mu,0}$ in the secret key, as $\mathbf{x}$ is undefined when sk is generated, and hence cannot invoke the simulation security of AKGS. Our second key idea is relying on a stronger security property of AKGS, named *piecewise security*, to hardwire simulated labels into the secret key in a piecemeal fashion.

Piecewise security of AKGS requires the following two properties: *i*) reverse sampleability — there is an efficient procedure $\mathsf{RevSamp}$ that can *perfectly* reversely sample the first label ℓ_1 given the output $\alpha f(\mathbf{x}) + \beta$ and all the other labels $\ell_2, \ldots, \ell_m$, and *ii*) marginal randomness — each ℓ_j of the following labels for $j > 1$ is uniformly distributed over $\mathbb{Z}_p$ even given all subsequent *label functions* $\mathbf{L}_{j+1}, \ldots, \mathbf{L}_m$. More formally,

$$\left\{\ell_1 \leftarrow \langle \mathbf{L}_1, (1, \mathbf{x})\rangle, \quad \mathbf{L}_2, \ldots, \mathbf{L}_m\right\} \equiv \left\{\ell_1' \xleftarrow{\$} \mathsf{RevSamp}(\cdots), \quad \mathbf{L}_2, \ldots, \mathbf{L}_m\right\}, \quad (2)$$

$$\left\{\ell_j \leftarrow \langle \mathbf{L}_j, (1, \mathbf{x})\rangle, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m\right\} \equiv \left\{\ell_j' \xleftarrow{\$} \mathbb{Z}_p \quad , \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m\right\}. \quad (3)$$

In Equation (2), $\ell_1' \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m)$. These properties are natural and satisfied by existing AKGS for ABPs and arithmetic formulae [11, 37].

Adaptive Security via Piecewise Security. We are now ready to prove adaptive security of our 1-ABE. The proof strategy is to first hardwire ℓ_1 in the ciphertext and sample it reversely as $\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, 0, \ell_2, \ldots, \ell_m)$, where $\ell_j = \langle \mathbf{L}_j, (1, \mathbf{x})\rangle$ for $j > 1$ and $\mu f(\mathbf{x}) = 0$ by the constraint, as described in hybrid $k = 1$ below. The indistinguishability follows immediately from the function-hiding property of IPFE and the reverse sampleability of AKGS. Then, we gradually replace each remaining label function $\mathbf{L}_j$ for $j > 1$ with a randomly sampled label $\ell_j \xleftarrow{\$} \mathbb{Z}_p$ in the secret key, as described in hybrids $1 \le k \le m+1$. It is easy to observe that in the final hybrid $k = m + 1$, where all labels $\ell_2, \ldots, \ell_m$ are random and ℓ_1 reversely sampled *without* μ, the value μ is information-theoretically hidden.

HYBRID $1 \le k \le m+1$	HYBRID $k:1$ OR $k:2$
$\mathsf{sk}(f, \mu)$: $\mathsf{isk}_1(\ \mathbf{0} \ \Vert\ 1 \ \Vert\ 0 \ \Vert\ 0\ )$	
$1 < j < k$: $\mathsf{isk}_j(\ \mathbf{0} \ \Vert\ 0 \ \Vert\ \ell_j \ \Vert\ 0\ )$	
$\mathsf{isk}_k(\ \mathbf{L}_k \ \Vert\ 0 \ \Vert\ 0 \ \Vert\ 0\ )$	$\mathsf{isk}_k(\ \mathbf{0} \ \Vert\ 0 \ \Vert\ 0 \ \Vert\ 1\ )$
$j > k$: $\mathsf{isk}_j(\ \mathbf{L}_j \ \Vert\ 0 \ \Vert\ 0 \ \Vert\ 0\ )$	
$\mathsf{ct}(\mathbf{x})$: $\mathsf{ict}(\ (1, \mathbf{x}) \ \Vert\ \ell_1 \ \Vert\ 1 \ \Vert\ 0\ )$	$\mathsf{ict}(\ (1, \mathbf{x}) \ \Vert\ \ell_1 \ \Vert\ 1 \ \Vert\ \ell_k\ )$
$\ell_1 \xleftarrow{\$} \mathsf{RevSamp}(\cdots)$, FOR $1 < j < k$: $\ell_j \xleftarrow{\$} \mathbb{Z}_p$	$\ell_k \leftarrow \langle \mathbf{L}_k, (1, \mathbf{x})\rangle$ OR $\ell_k \xleftarrow{\$} \mathbb{Z}_p$

To move from hybrid k to $k + 1$, we want to switch the k^{th} IPFE secret key isk_k from encoding the label function $\mathbf{L}_k$ to a simulated label $\ell_k \xleftarrow{\$} \mathbb{Z}_p$. This is possible via two moves. First, by the function-hiding property of IPFE, we can hardwire the honest $\ell_k = \langle \mathbf{L}_k, (1, \mathbf{x})\rangle$ in the ciphertext as in hybrid $k : 1$ (recall that at encryption time, $\mathbf{x}$ is known). Then, by the marginal randomness

property of AKGS, we switch to sample ℓ_k as random in hybrid $k:2$. Lastly, hybrid $k:2$ is indistinguishable to hybrid $k+1$ again by the function-hiding property of IPFE.

1-ABE for ABPs. Plugging in the AKGS for ABPs by Ishai and Wee [37], we immediately obtain 1-ABE for ABPs based on k-Lin. The size of the garbling grows linearly with the number of vertices $|V|$ in the graph describing the ABP, i.e., $m = O(|V|)$. Combined with the fact that IPFE has linear-size secret keys and ciphertexts, our 1-ABE scheme for ABPs has secret keys of size $O(m|\mathbf{x}|) = O(|V||\mathbf{x}|)$ and ciphertexts of size $O(|\mathbf{x}|)$. This gives an efficiency improvement over previous 1-ABE or ABE schemes for ABPs [23,37], where the secret key size grows linearly with the number of edges $|E|$ in the ABP graph, due to that their schemes first convert ABPs into an arithmetic span program, which incurs the efficiency loss.

Discussion. Our method for constructing 1-ABE is generic and modular. In particular, it has the advantage that the proof of adaptive security is agnostic of the computation being performed and merely carries out the simulation of AKGS in a mechanic way. Indeed, if we plug in an AKGS for arithmetic formulae or any other classes of non-uniform computation, the proof remains the same. (Our 1-ABE for logspace Turing machines also follows the same blueprint, but needs additional ideas.) Furthermore, note that our method departs from the partial selectivization technique used in [44], which is not applicable to arithmetic computation as the security reduction cannot afford to guess even one component of the input $\mathbf{x}$. The problem is circumvented by using IPFE to hardwire the labels (i.e., ℓ_1, ℓ_k) that depend on $\mathbf{x}$ in the ciphertext.

2.2 Full-Fledged ABE via IPFE

From 1-ABE for the 1-key 1-ciphertext setting to full-fledged ABE, we need to support publishing multiple keys and make encryption public-key. It turns out that the security of our 1-ABE scheme directly extends to the many-key 1-ciphertext (still secret-key) setting via a simple hybrid argument. Consider the scenario where a ciphertext ct and multiple keys $\{\mathsf{sk}_q(f_q, \mu_q)\}_{q\in[Q]}$ that are unauthorized to decrypt the ciphertext are published. Combining the above security proof for 1-ABE with a hybrid argument, we can gradually switch each secret key sk_q from encoding honest label functions encapsulating μ_q to ones encapsulating an independent secret $\mu'_q \xleftarrow{\$} \mathbb{Z}_p$. Therefore, all the secrets $\{\mu_q\}_{q\in[Q]}$ are hidden.

The security of our 1-ABE breaks down once two ciphertexts are released. Consider publishing just a single secret key $\mathsf{sk}(f, \mu)$ and two ciphertexts $\mathsf{ct}_1(\mathbf{x}_1)$, $\mathsf{ct}_2(\mathbf{x}_2)$. Since the label functions $L_1, \ldots, L_m$ are encoded in sk, decryption computes two AKGS garblings $\widehat{f(\mathbf{x}_1)}_{\mu,0}$ and $\widehat{f(\mathbf{x}_2)}_{\mu,0}$ generated using the *same* label functions. However, AKGS security does not apply when the label functions are reused.

What we wish is that IPFE decryption computes two garblings $\widehat{f(\mathbf{x}_1)}_{\mu,0} = (L_1(\mathbf{x}_1), \ldots, L_m(\mathbf{x}_1))$ and $\widehat{f(\mathbf{x}_2)}_{\mu,0} = (L'_1(\mathbf{x}_2), \ldots, L'_m(\mathbf{x}_2))$ using *independent*

label functions. This can be achieved in a *computational* fashion relying on the fact that the IPFE scheme encodes the vectors and the decryption results in the exponent of bilinear pairing groups. Hence we can rely on computational assumptions such as SXDH or k-Lin, combined with the function-hiding property of IPFE to argue that the produced garblings are computationally independent. We modify the 1-ABE scheme as follows:

– If SXDH holds in the pairing groups, we encode in the ciphertext $(1, \mathbf{x})$ multiplied by a random scalar $s \xleftarrow{\$} \mathbb{Z}_p$. As such, decryption computes $(sL_1(\mathbf{x}), \ldots, sL_m(\mathbf{x}))$ *in the exponent*. We argue that the label functions $sL_1, \ldots, sL_m$ are computationally random *in the exponent*: By the function-hiding property of IPFE, it is indistinguishable to multiply s not with the ciphertext vector, but with the coefficient vectors in the secret key as depicted below on the right; by DDH (in G_2) and the linearity of Garble (i.e., the coefficients $\mathbf{L}_j$ depend linearly on the secrets α, β and the randomness $\mathbf{r}$ used by Garble), $s\mathbf{L}_j$ are the coefficients of pseudorandom label functions.

Algorithms based on SXDH		Hybrid	
			$\approx \mathbf{L}'_j$ (fresh)
$\mathsf{sk}(f, \mu)$:	$j \in [m]$: $\mathsf{isk}_j($ $\mathbf{L}_j$ $\Vert$ $\mathbf{0}$ $)$	$\mathsf{isk}_j($ $\mathbf{L}_j$ $\Vert$ $s\mathbf{L}_j$ $\Vert$ $\mathbf{0}$ $)$	
$\mathsf{ct}(\mathbf{x})$:	$\mathsf{ict}($ $s(1, \mathbf{x})$ $\Vert$ $\mathbf{0}$ $)$	$\mathsf{ict}($ $\mathbf{0}$ $\Vert$ $(1, \mathbf{x})$ $\Vert$ $\mathbf{0}$ $)$	

– If k-Lin holds in the pairing groups, we encode in the secret key k independent copies of label functions $L_1^t, \ldots, L_m^t$ for $t \in [k]$, and in the ciphertexts k copies of $(1, \mathbf{x})$ multiplied with independent random scalars $\mathbf{s}[t]$ for $t \in [k]$. This way, decryption computes a random linear combination of the garblings $(\sum_{t\in[k]} \mathbf{s}[t]L_1^t(\mathbf{x}), \ldots, \sum_{t\in[k]} \mathbf{s}[t]L_m^t(\mathbf{x}))$ in the exponent, which via a similar hybrid as above corresponds to pseudorandom label functions in the exponent.

Algorithms based on k-Lin

$\mathsf{sk}(f, \mu)$:	$j \in [m]$: $\mathsf{isk}_j($	$\mathbf{L}_j^1 \Vert \cdots \Vert \mathbf{L}_j^k \Vert \mathbf{0}$ $)$
$\mathsf{ct}(\mathbf{x})$:	$\mathsf{ict}($	$\mathbf{s}[1](1, \mathbf{x}) \Vert \cdots \Vert \mathbf{s}[k](1, \mathbf{x}) \Vert \mathbf{0}$ $)$

Hybrid

		$\approx \mathbf{L}'_j$ (fresh)
$\mathsf{sk}(f, \mu)$:	$j \in [m]$: $\mathsf{isk}_j($	$\mathbf{L}_j^1 \Vert \cdots \Vert \mathbf{L}_j^k \Vert \sum_{t\in[k]} \mathbf{s}[t]\mathbf{L}_j^t \Vert \mathbf{0}$ $)$
$\mathsf{ct}(\mathbf{x})$:	$\mathsf{ict}($	$\mathbf{0} \Vert \cdots \Vert \mathbf{0} \Vert (1, \mathbf{x}) \Vert \mathbf{0}$ $)$

The above modification yields a secret-key ABE secure in the many-ciphertext many-key setting. The final hurdle is how to make the scheme public-key, which we resolve using slotted IPFE.

Slotted IPFE. Proposed in [50], *slotted* IPFE is a hybrid between a secret-key function-hiding IPFE and a public-key IPFE. Here, a vector $\mathbf{u} \in \mathbb{Z}_p^n$ is divided into two parts $(\mathbf{u}_{\text{pub}}, \mathbf{u}_{\text{priv}})$ with $\mathbf{u}_{\text{pub}} \in \mathbb{Z}_p^{n_{\text{pub}}}$ in the public slot and $\mathbf{u}_{\text{priv}} \in \mathbb{Z}_p^{n_{\text{priv}}}$ in the private slot ($n_{\text{pub}} + n_{\text{priv}} = n$). Like a usual secret-key IPFE,

the encryption algorithm $\mathsf{IPFE.Enc}$ using the master secret key msk can encrypt to both the public and private slots, i.e., encrypting any vector $\mathbf{u}$. In addition, there is an $\mathsf{IPFE.SlotEnc}$ algorithm that uses the master public key mpk, but can only encrypt to the public slot, i.e., encrypting vectors such that $\mathbf{u}_{\mathrm{priv}} = \mathbf{0}$. Since anyone can encrypt to the public slot, it is impossible to hide the public slot part $\mathbf{v}_{\mathrm{pub}}$ of a secret-key vector $\mathbf{v}$. As a result, slotted IPFE guarantees function-hiding only w.r.t. the private slot, and the weaker indistinguishability security w.r.t. the public slot. Based on the construction of slotted IPFE in [49], we obtain adaptively secure slotted IPFE based on k-Lin.

The aforementioned secret-key ABE scheme can be easily turned into a public-key one with slotted IPFE: The ABE encryption algorithm simply uses $\mathsf{IPFE.SlotEnc}$ and mpk to encrypt to the public slots. In the security proof, we move vectors encrypted in the public slot of the challenge ciphertext to the private slot, where function-hiding holds and the same security arguments outlined above can be carried out.

Discussion. Our method can be viewed as using IPFE to implement dual system encryption [57]. We believe that IPFE provides a valuable abstraction, making it conceptually simpler to design strategies for moving information between the secret key and the ciphertext, as done in the proof of 1-ABE, and for generating independent randomness, as done in the proof of full ABE. The benefit of this abstraction is even more prominent when it comes to ABE for logspace Turing machines.

2.3 1-ABE for Logspace Turing Machines

We now present ideas for constructing 1-ABE for L, and then its extension to NL and how to handle DFA and NFA as special cases for better efficiency. Moving to full-fledged ABE follows the same ideas in the previous subsection, though slightly more complicated, which we omit in this overview.

1-ABE for L enables generating a single secret key $\mathsf{sk}(M, \mu)$ for a Turing machine M and secret μ, and a ciphertext $\mathsf{ct}(\mathbf{x}, T, S)$ specifying an input $\mathbf{x}$ of length N, a polynomial time bound $T = \mathrm{poly}(N)$, and a logarithmic space bound $S = O(\log N)$ such that decryption reveals $\mu M|_{N,T,S}(\mathbf{x})$, where $M|_{N,T,S}(\mathbf{x})$ represents the computation of running $M(\mathbf{x})$ for T steps with a work tape of size S, which outputs 1 if and only if the computation lands in an accepting state after T steps and has *never* exceeded the space bound S. A key feature of ABE for uniform computation is that a secret key $\mathsf{sk}(M, \mu)$ can decrypt ciphertexts with inputs of unbounded lengths and unbounded time / (logarithmic) space bounds. (In contrast, for non-uniform computation, the secret key decides the input length and time/space bounds.) Our 1-ABE for L follows the same blueprint of combining AKGS with IPFE, but uses new ideas in order to implement the unique feature of ABE for uniform computation.

Notations for Turing Machines. We start with introducing notations for logspace Turing machines (TM) over the binary alphabet. A TM $M = (Q, q_{\mathrm{acc}}, \delta)$

consists of Q states, with the initial state being 1 and an accepting state[9] $q_{\text{acc}} \in [Q]$, and a transition function δ. The computation of $M|_{N,T,S}(\mathbf{x})$ goes through a sequence of $T+1$ *configurations* $(\mathbf{x}, (i, j, \mathbf{W}, q))$, where $i \in [N]$ is the input tape pointer, $j \in [S]$ the work tape pointer, $\mathbf{W} \in \{0,1\}^S$ the content of the work tape, and $q \in [Q]$ the state. The initial *internal* configuration is thus $(i = 1, j = 1, \mathbf{W} = \mathbf{0}_S, q = 1)$, and the transition from one internal configuration $(i, j, \mathbf{W}, q)$ to the next $(i', j', \mathbf{W}', q')$ is governed by the transition function δ and the input $\mathbf{x}$. Namely, if $\delta(q, \mathbf{x}[i], \mathbf{W}[j]) = (q', w', \Delta i, \Delta j)$,

$$(i, j, \mathbf{W}, q) \to (i' = i + \Delta i,\ j' = j + \Delta j,\ \mathbf{W}' = \mathsf{overwrite}(\mathbf{W}, j, w'),\ q').$$

In other words, the transition function δ on input state q and bits $\mathbf{x}[i]$, $\mathbf{W}[j]$ on the input and work tape under scan, outputs the next state q', the new bit $w' \in \{0,1\}$ to be written to the work tape, and the directions $\Delta i, \Delta j \in \{0, \pm 1\}$ to move the input and work tape pointers. The next internal configuration is then derived by updating the current configuration accordingly, where $\mathbf{W}' = \mathsf{overwrite}(\mathbf{W}, j, w')$ is a vector obtained by overwriting the j^{th} cell of $\mathbf{W}$ with w' and keeping the other cells unchanged.

AKGS for Logspace Turing Machines. To obtain an AKGS for L, we represent the TM computation algebraically as a sequence of matrix multiplications over $\mathbb{Z}_p$, for which we design an AKGS. To do so, we represent each internal configuration as a basis vector $\mathbf{e}_{(i,j,\mathbf{W},q)}$ of dimension $NS2^S Q$ with a single 1 at position $(i, j, \mathbf{W}, q)$. We want to find a *transition matrix* $\mathbf{M}(\mathbf{x})$ (depending on δ and $\mathbf{x}$) such that moving to the next state $\mathbf{e}_{(i',j',\mathbf{W}',q')}$ simply involves (right) multiplying $\mathbf{M}(\mathbf{x})$, i.e., $\mathbf{e}^\intercal_{(i,j,\mathbf{W},q)}\mathbf{M}(\mathbf{x}) = \mathbf{e}^\intercal_{(i',j',\mathbf{W}',q')}$. It is easy to verify that the correct transition matrix is

$$\begin{aligned}
\mathbf{M}(\mathbf{x})[(i, j, \mathbf{W}, q), (i', j', \mathbf{W}', q')] &= \mathsf{CanTransit}[(i, j, \mathbf{W}), (i', j', \mathbf{W}')] \\
&\quad \times \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],i'-i,j'-j}[q, q'], \qquad (4)\\
\mathsf{CanTransit}[(i, j, \mathbf{W}), (i', j', \mathbf{W}')] = 1 &\quad \text{iff } \mathbf{W}'[\neq j] = \mathbf{W}[\neq j] \text{ and} \\
&\qquad\qquad i' - i, j' - j \in \{0, \pm 1\}, \\
\mathbf{M}_{x,w,w',\Delta i,\Delta j}[q, q'] = 1 &\quad \text{iff } \delta(q, x, w') = (q', w', \Delta i, \Delta j). \qquad (5)
\end{aligned}$$

Here, $\mathsf{CanTransit}[(i, j, \mathbf{W}), (i'j', \mathbf{W}')]$ indicates whether it is possible, irrespective of δ, to move from an internal configuration with $(i, j, \mathbf{W})$ to one with $(i', j', \mathbf{W}')$. If possible, then $\mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],\Delta i,\Delta j}[q, q']$ indicates whether δ permits moving from state q with current read bits $x = \mathbf{x}[i], w = \mathbf{W}[j]$ to state q' with overwriting bit $w' = \mathbf{W}'[j]$ and moving directions $\Delta i = i' - i, \Delta j = j' - j$. Armed with this, the TM computation can be done by right multiplying the matrix $\mathbf{M}(\mathbf{x})$ for T times with the initial configuration $\mathbf{e}^\intercal_{(1,1,\mathbf{0},1)}$, reaching the final configuration $\mathbf{e}^\intercal_{(i_T,j_T,\mathbf{W}_T,q_T)}$, and then testing whether $q_T = q_{\text{acc}}$. More precisely,

$$M|_{N,T,S}(\mathbf{x}) = \mathbf{e}^\intercal_{(1,1,\mathbf{0},1)}\big(\mathbf{M}(\mathbf{x})\big)^T\mathbf{t} \text{ for } \mathbf{t} = \mathbf{1}_{NS2^S} \otimes \mathbf{e}_{q_{\text{acc}}}.$$

[9] For simplicity, in this overview, we assume there is only one accepting state.

To construct AKGS for L, it boils down to construct AKGS for matrix multiplication. Our construction is inspired by the randomized encoding for arithmetic NC^1 scheme of [11] and the garbling mechanism for multiplication gates in [19]. Let us focus on garbling the computation $M|_{N,T,S}(\mathbf{x})$ with secrets $\alpha = \mu$ and $\beta = 0$ (the case needed in our 1-ABE). The garbling algorithm Garble produces the following affine label functions of $\mathbf{x}$:

$$\begin{aligned} \ell_{\text{init}} &= L_{\text{init}}(\mathbf{x}) = \mathbf{e}^\intercal_{(1,1,\mathbf{0},1)}\mathbf{r}_0, \\ t \in [T]: \quad \boldsymbol{\ell}_t &= (\ell_{t,z}) = \big(L_{t,z}(\mathbf{x})\big)_z = -\boxed{\mathbf{r}_{t-1}} + \mathbf{M}(\mathbf{x})\mathbf{r}_t, \\ \boldsymbol{\ell}_{T+1} &= (\ell_{T+1,z})_z = \big(L_{T+1,z}(\mathbf{x})\big)_z = -\boxed{\mathbf{r}_T} + \mu\mathbf{t}. \end{aligned}$$

Here, $z = (i, j, \mathbf{W}, q)$ runs through all $NS2^SQ$ possible internal configurations and $\mathbf{r}_t \xleftarrow{\$} \mathbb{Z}_p^{[N]\times[S]\times\{0,1\}^S\times[Q]}$. The evaluation proceeds inductively, starting with $\ell_{\text{init}} = \mathbf{e}^\intercal_{(1,1,\mathbf{0},1)}\mathbf{r}_0$, going through $\mathbf{e}^\intercal_{(i_t,j_t,\mathbf{W}_t,q_t)}\mathbf{r}_t$ for every $t \in [T]$ using the identity below, and completing after T steps by combining $\mathbf{e}^\intercal_{(i_T,j_T,\mathbf{W}_T,q_T)}\mathbf{r}_T$ with $\boldsymbol{\ell}_{T+1}$ to get $\mathbf{e}^\intercal_{(i_T,j_T,\mathbf{W}_T,q_T)}\mu\mathbf{t} = \mu M|_{N,T,S}(\mathbf{x})$ as desired:

$$\mathbf{e}^\intercal_{(i_{t+1},j_{t+1},\mathbf{W}_{t+1},q_{t+1})}\mathbf{r}_{t+1} = \mathbf{e}^\intercal_{(i_t,j_t,\mathbf{W}_t,q_t)}\mathbf{r}_t + \mathbf{e}^\intercal_{(i_t,j_t,\mathbf{W}_t,q_t)}\underbrace{(-\mathbf{r}_t + \mathbf{M}(\mathbf{x})\mathbf{r}_{t+1})}_{\boldsymbol{\ell}_{t+1}}.$$

We now show that the above AKGS is *piecewise secure*. First, ℓ_{init} is reversely sampleable. Since Eval is linear in the labels and ℓ_{init} has coefficient 1, given all but the first label ℓ_{init}, one can reversely sample ℓ_{init}, the value uniquely determined by the linear equation[10] imposed by the correctness of Eval. Second, the marginal randomness property holds because every label $\boldsymbol{\ell}_t$ is random due to the random additive term $\mathbf{r}_{t-1}$ that is not used in subsequent label functions $L_{t',z}$ for all $t' > t$ and z, nor in the non-constant terms of $L_{t,z}$'s — we call $\mathbf{r}_{t-1}$ the *randomizers* of $\boldsymbol{\ell}_t$ (highlighted in the box). Lastly, we observe that the size of the garbling is $(T+1)NS2^SQ + 1$.

1-ABE for L. We now try to construct 1-ABE for L from AKGS for L, following the same blueprint of using IPFE. Yet, applying the exact same method for non-uniform computation fails for multiple reasons. In 1-ABE for non-uniform computation, the ciphertext ct contains a single IPFE ciphertext ict encoding $(1, \mathbf{x})$, and the secret key sk contains a set of IPFE secret keys isk_j encoding all the label functions. However, in the uniform setting, the secret key $\mathsf{sk}(M, \mu)$ depends only on the TM M and the secret μ, and is supposed to work with ciphertexts $\mathsf{ct}(\mathbf{x}, T, S)$ with unbounded $N = |\mathbf{x}|, T, S$. Therefore, at key generation time, the size of the AKGS garbling, $(T+1)NS2^SQ + 1$, is *unknown*, let alone generating and encoding all the label functions. Moreover, we want our 1-ABE to be compact, with secret key size $|\mathsf{sk}| = O(Q)$ linear in the number Q of states and ciphertext size $|\mathsf{ct}| = O(TNS2^S)$ (ignoring polynomial factors in the

[10] This means $\mathsf{RevSamp}$ is deterministic, and we can reversely sample ℓ_{init} *in the exponent* and when the randomness is not uniform, which is important for our construction.

security parameter). The total size of secret key and ciphertext is much smaller than the total number of label functions, i.e., $|\mathsf{sk}| + |\mathsf{ct}| \ll (T+1)NS2^S Q + 1$.

To overcome these challenges, our idea is that instead of encoding the label functions in the secret key or the ciphertext (for which there is not enough space), we let the secret key and the ciphertext jointly generate the label functions. For this idea to work, the label functions cannot be generated with true randomness which cannot be "compressed", and must use pseudorandomness instead. More specifically, our 1-ABE secret key $\mathsf{sk}(M, \mu)$ contains $\sim Q$ IPFE secret keys $\{\mathsf{isk}(\mathbf{v}_j)\}_j$, while the ciphertext $\mathsf{ct}(\mathbf{x}, T, S)$ contains $\sim TNS2^S$ IPFE ciphertexts $\{\mathsf{ict}(\mathbf{u}_i)\}_i$, such that decryption computes in the exponent $\sim TNS2^S Q$ cross inner products $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$ that correspond to a garbling of $M|_{N,T,S}(\mathbf{x})$ with secret μ. To achieve this, we rely crucially on the special *block structure* of the transition matrix $\mathbf{M}$ (which in turn stems from the structure of TM computation, where the same transition function is applied in every step). Furthermore, as discussed above, we replace every truly random value $\mathbf{r}_t[i, j, \mathbf{W}, q]$ with a product $\mathbf{r}_{\mathrm{x}}[t, i, j, \mathbf{W}]\mathbf{r}_{\mathrm{f}}[q]$, which can be shown pseudorandom in the exponent based on SXDH.[11]

Block Structure of the Transition Matrix. Let us examine the transition matrix again (cf. Equations (4) and (5)):

$$\mathbf{M}(\mathbf{x})[(i, j, \mathbf{W}, q), (i', j', \mathbf{W}', q')] = \mathsf{CanTransit}[(i, j, \mathbf{W}), (i', j', \mathbf{W}')] \times \mathbf{M}_{\mathbf{x}[i], \mathbf{W}[j], \mathbf{W}'[j], i'-i, j'-j}[q, q'].$$

We see that that every block $\mathbf{M}(\mathbf{x})[(i, j, \mathbf{W}, \textcolor{orange}{\sqcup}), (i', j', \mathbf{W}', \textcolor{orange}{\sqcup})]$ either is the $Q \times Q$ zero matrix or belongs to a small set $\mathcal{T}$ of a constant number of *transition blocks*:

$$\mathcal{T} = \left\{ \mathbf{M}_{x, w, w', \Delta i, \Delta j} \;\middle|\; x, w, w' \in \{0, 1\},\ \Delta i, \Delta j \in \{0, \pm 1\} \right\}.$$

Moreover, in the $\mathfrak{i} = (i, j, \mathbf{W})^{\text{th}}$ "block row", $\mathbf{M}(\mathbf{x})[(\mathfrak{i}, \textcolor{orange}{\sqcup}), (\textcolor{orange}{\sqcup}, \textcolor{orange}{\sqcup}, \textcolor{orange}{\sqcup}, \textcolor{orange}{\sqcup})]$, each transition block $\mathbf{M}_{x, w, w', \Delta i, \Delta j}$ either does not appear at all if $x \neq \mathbf{x}[i]$ or $w' \neq \mathbf{W}[j]$, or appears once as the block $\mathbf{M}(\mathbf{x})[(\mathfrak{i}, \textcolor{orange}{\sqcup}), (\mathfrak{i}', \textcolor{orange}{\sqcup})]$, where $\mathfrak{i}'$ is the triplet obtained by updating $\mathfrak{i}$ appropriately according to $(w', \Delta i, \Delta j)$:

$$\mathfrak{i}' \stackrel{\text{def}}{=\!=} \mathfrak{i} \boxplus (w', \Delta i, \Delta j) = (i + \Delta i,\ j + \Delta j,\ \mathbf{W}' = \mathsf{overwrite}(\mathbf{W}, j, w')),$$
$$\mathbf{M}(\mathbf{x})[(\mathfrak{i}, \textcolor{orange}{\sqcup}), (\mathfrak{i}', \textcolor{orange}{\sqcup})] = \mathbf{M}_{\mathbf{x}[i], \mathbf{W}[j], w', \Delta i, \Delta j}.$$

Thus we can "decompose" every label $\boldsymbol{\ell}_t[\mathfrak{i}, q]$ as an inner product $\langle \mathbf{u}_{t,\mathfrak{i}}, \mathbf{v}_q \rangle$ as

$$\begin{aligned}
\boldsymbol{\ell}_t[\mathfrak{i}, q] &= -\mathbf{r}_{t-1}[\mathfrak{i}, q] + \mathbf{M}(\mathbf{x})[(\mathfrak{i}, q)(\textcolor{orange}{\sqcup}, \textcolor{orange}{\sqcup}, \textcolor{orange}{\sqcup}, \textcolor{orange}{\sqcup})]\mathbf{r}_t \\
&= -\mathbf{r}_{t-1}[\mathfrak{i}, q] + \sum_{w', \Delta i, \Delta j} \big(\mathbf{M}_{\mathbf{x}[i], \mathbf{W}[j], w', \Delta i, \Delta j} \mathbf{r}_t[\mathfrak{i}', \textcolor{orange}{\sqcup}]\big)[q] \quad \big(\mathfrak{i}' = \mathfrak{i} \boxplus (w', \Delta i, \Delta j)\big) \\
&= -\mathbf{r}_{\mathrm{x}}[t-1, \mathfrak{i}]\mathbf{r}_{\mathrm{f}}[q] + \sum_{w', \Delta i, \Delta j} \mathbf{r}_{\mathrm{x}}[t, \mathfrak{i}'] \big(\mathbf{M}_{\mathbf{x}[i], \mathbf{W}[j], w', \Delta i, \Delta j} \mathbf{r}_{\mathrm{f}}\big)[q] \\
&= \langle \mathbf{u}_{t,\mathfrak{i}}, \mathbf{v}_q \rangle, \qquad \big(\mathbf{r}_{t''}[\mathfrak{i}'', q''] = \mathbf{r}_{\mathrm{x}}[t'', \mathfrak{i}'']\mathbf{r}_{\mathrm{f}}[q'']\big)
\end{aligned}$$

[11] Our scheme readily extends to be based on k-Lin. However, that makes the scheme more complex to present. We choose to present this scheme using SXDH in this paper.

where vectors $\mathbf{u}_{t,\mathsf{i}}$ and $\mathbf{v}_q$ are as follows, with $\mathbb{1}\{\cdots\}$ indicating if the conditions (its argument) are true:

$$\begin{aligned}\mathbf{u}_{t,\mathsf{i}} &= (\ \mathbf{r}_{\mathrm{x}}[t-1,\mathsf{i}] \parallel \cdots \parallel \mathbf{r}_{\mathrm{x}}[t,\mathsf{i}'] \cdot \mathbb{1}\{x = \mathbf{x}[i], w = \mathbf{W}[j]\} \parallel \cdots \parallel \mathbf{0}),\\ \mathbf{v}_q &= (\quad -\mathbf{r}_{\mathrm{f}}[q] \quad \parallel \cdots \parallel \quad (\mathbf{M}_{x,w,w',\Delta i,\Delta j}\mathbf{r}_{\mathrm{f}})[q] \quad \parallel \cdots \parallel \mathbf{0}).\end{aligned}$$

Similarly, we can "decompose" $\ell_{\mathrm{init}} = \mathbf{e}_{1,1,\mathbf{0},1}^{\intercal}\mathbf{r}_0$ as $\langle \mathbf{r}_{\mathrm{x}}[0,1,1,\mathbf{0}], \mathbf{r}_{\mathrm{f}}[1]\rangle$. (For simplicity in the discussion below, we omit details on how to handle $\boldsymbol{\ell}_{T+1}$.) Given such decomposition, our semi-compact 1-ABE scheme follows immediately by using IPFE to compute the garbling:

Honest Algorithms

$$\begin{aligned}\mathsf{sk}(M,\mu)\colon \mathsf{isk}_{\mathrm{init}}(\quad \mathbf{r}_{\mathrm{f}}[1] \quad \parallel \mathbf{0}\ ), \quad \forall q\colon \quad \mathsf{isk}_q(\ \mathbf{u}_{t,\mathsf{i}} \parallel \mathbf{0}\ )\\ \mathsf{ct}(\mathbf{x},T,S)\colon \mathsf{ict}_{\mathrm{init}}(\ \mathbf{r}_{\mathrm{x}}[0,1,1,\mathbf{0}] \parallel \mathbf{0}\ ), \quad \forall t,\mathsf{i}\colon \mathsf{ict}_{i,\mathsf{i}}(\ \mathbf{v}_q \parallel \mathbf{0}\ )\end{aligned}$$

Decrypting the pair $\mathsf{isk}_{\mathrm{init}}, \mathsf{ict}_{\mathrm{init}}$ (generated using one master secret key) gives exactly the first label ℓ_{init}, while decrypting $\mathsf{isk}_q, \mathsf{ict}_{t,\mathsf{i}}$ (generated using another master secret key) gives the label $\boldsymbol{\ell}_t[\mathsf{i},q]$ in the exponent, generated using pseudorandomness $\mathbf{r}_t[\mathsf{i},q] = \mathbf{r}_{\mathrm{x}}[t,\mathsf{i}]\mathbf{r}_{\mathrm{f}}[q]$. Note that the honest algorithms encode $\mathbf{r}_{\mathrm{f}}[q]$ (in $\mathbf{v}_q$) and $\mathbf{r}_{\mathrm{x}}[t,\mathsf{i}]$ (in $\mathbf{u}_{t,\mathsf{i}}$) in IPFE secret keys and ciphertexts that use the *two* source groups G_1 and G_2 respectively. As such, we cannot directly use the SXDH assumption to argue the pseudorandomness of $\mathbf{r}_t[\mathsf{i},q]$. In the security proof, we will use the function-hiding property of IPFE to move both $\mathbf{r}_{\mathrm{x}}[t,\mathsf{i}]$ and $\mathbf{r}_{\mathrm{f}}[q]$ into the same source group before invoking SXDH.

Adaptive Security. To show adaptive security, we follow the same blueprint of going through a sequence of hybrids, where we first hardcode ℓ_{init} and sample it reversely using $\mathsf{RevSamp}$, and next simulate the other labels $\boldsymbol{\ell}_t[\mathsf{i},q]$ one by one. Hardwiring ℓ_{init} is easy by relying on the function-hiding property of IPFE. However, it is now more difficult to simulate $\boldsymbol{\ell}_t[\mathsf{i},q]$ because *i*) before simulating $\boldsymbol{\ell}_t[\mathsf{i},q]$, we need to switch its *randomizer* $\mathbf{r}_{t-1}[\mathsf{i},q] = \mathbf{r}_{\mathrm{x}}[t-1,\mathsf{i}]\mathbf{r}_{\mathrm{f}}[q]$ to truly random $\mathbf{r}_{t-1}[\mathsf{i},q] \xleftarrow{\$} \mathbb{Z}_p$, which enables us to simulate the label $\boldsymbol{\ell}_t[\mathsf{i},q]$ as random; and *ii*) to keep simulation progressing, we need to switch the random $\boldsymbol{\ell}_t[\mathsf{i},q]$ back to a pseudorandom value $\boldsymbol{\ell}_t[\mathsf{i},q] = \mathbf{s}_{\mathrm{x}}[t,\mathsf{i}]\mathbf{s}_{\mathrm{f}}[q]$, as otherwise, there is not enough space to store all $\sim TNS2^S Q$ random labels $\boldsymbol{\ell}_t[\mathsf{i},q]$.

We illustrate how to carry out above proof steps in the simpler case where the the adversary queries for the ciphertext first and the secret key second. The other case where the secret key is queried first is handled using similar ideas, but the technicality becomes much more delicate.

In hybrid (t,i), the first label ℓ_{init} is reversely sampled and hardcoded in the secret key $\mathsf{isk}_{\mathrm{init}}$, i.e., $\mathsf{ict}_{\mathrm{init}}$ encrypts $(1 \parallel 0)$ and $\mathsf{isk}_{\mathrm{init}}$ encrypts $(\ell_{\mathrm{init}} \parallel 0)$ with $\ell_{\mathrm{init}} \leftarrow \mathsf{RevSamp}(\cdots)$. All labels $\boldsymbol{\ell}_{t'}[\mathsf{i}',q]$ with $(t',\mathsf{i}') < (t,\mathsf{i})$ have been simulated as $\mathbf{s}_{\mathrm{x}}[t',\mathsf{i}']\mathbf{s}_{\mathrm{f}}[q]$ — observe that the ciphertext $\mathsf{ict}_{t',\mathsf{i}'}$ encodes only $\mathbf{s}_{\mathrm{f}}[t',\mathsf{i}']$ in the second slot, which is multiplied by $\mathbf{s}_{\mathrm{f}}[q]$ in the second slot of isk_q. On the other hand, all labels $\boldsymbol{\ell}_{t'}[\mathsf{i}',q]$ with $(t',\mathsf{i}') \geq (t,\mathsf{i})$ are generated honestly as the honest algorithms do.

$$
\begin{array}{rl}
 & \textsc{Hybrid}\ (t,\mathsf{i}),\ \boxed{(t,\mathsf{i}):1}\ \textsc{and}\ \boxed{(t,\mathsf{i})+1} \\
\mathsf{ct}(\mathbf{x},T,S): & (t',\mathsf{i}')<(t,\mathsf{i}):\ \mathsf{ict}_{t',\mathsf{i}'}(\ \mathbf{0}\ \|\ \mathbf{s}_{\mathrm{x}}[t',\mathsf{i}']\ \|\ 0\ ) \\
 & (t',\mathsf{i}')=(t,\mathsf{i}):\ \mathsf{ict}_{t,\mathsf{i}}(\ \mathbf{u}_{t,\mathsf{i}}\ \boxed{\mathbf{0}}\ \boxed{\mathbf{0}}\ \|\ 0\ \boxed{0}\ \boxed{\mathbf{s}_{\mathrm{x}}[t,\mathsf{i}]}\ \|\ 0\ \boxed{1}\ \boxed{0}\ ) \\
 & (t',\mathsf{i}')>(t,\mathsf{i}):\ \mathsf{ict}_{t',\mathsf{i}'}(\ \mathbf{u}_{t',\mathsf{i}'}\ \|\ 0\ \|\ 0\ ) \\
\mathsf{sk}(M,\mu): & q\in[Q]:\quad \mathsf{isk}_q(\ \mathbf{v}_q\ \|\ \mathbf{s}_{\mathrm{f}}[q]\ \|\ 0\ \boxed{\boldsymbol{\ell}_t[\mathsf{i},q]}\ \boxed{0}\ )
\end{array}
$$

Moving from hybrid (t,i) to its successor $(t,\mathsf{i})+1$, the only difference is that labels $\boldsymbol{\ell}_t[\mathsf{i},q]$ are switched from being honestly generated $\langle \mathbf{u}_{t,\mathsf{i}}, \mathbf{v}_q\rangle$ to pseudorandom $\mathbf{s}_{\mathrm{x}}[t,\mathsf{i}]\mathbf{s}_{\mathrm{f}}[q]$, as depicted above with values in the solid line box (the rest of the hybrid is identical to hybrid (t,i)). The transition can be done via an intermediate hybrid $(t,\mathsf{i}):1$ with values in the dash line box. In this hybrid, all labels $\boldsymbol{\ell}_t[\mathsf{i},q]$ produced as inner products of all $\mathbf{v}_q$'s and $\mathbf{u}_{t,\mathsf{i}}$ are temporarily hardcoded in the secret keys isk_q, using the third slot (which is zeroed out in all the other $\mathbf{u}_{(t',\mathsf{i}')\neq(t,\mathsf{i})}$'s). Furthermore, $\mathbf{u}_{t,\mathsf{i}}$ is removed from $\mathsf{ict}_{t,\mathsf{i}}$. As such, the random scalar $\mathbf{r}_{\mathrm{x}}[t-1,\mathsf{i}]$ (formerly embedded in $\mathbf{u}_{t,\mathsf{i}}$) no longer appears in the exponent of group G_1, and $\ell_{\mathrm{init}} \leftarrow \mathsf{RevSamp}(\cdots)$ can be performed using $\mathbf{r}_{\mathrm{x}}[t-1,\mathsf{i}], \mathbf{r}_{\mathrm{f}}[q], \mathbf{r}_{t-1}[\mathsf{i},q]$ in the exponent of G_2. Therefore, we can invoke the SXDH assumption in G_2 to switch the randomizers $\mathbf{r}_{t-1}[\mathsf{i},q] = \mathbf{r}_{\mathrm{x}}[t-1,\mathsf{i}]\mathbf{r}_{\mathrm{f}}[q]$ to be truly random, and hence so are the labels $\boldsymbol{\ell}_t[\mathsf{i},q] \xleftarrow{\$} \mathbb{Z}_p$. By a similar argument, this intermediate hybrid $(t,\mathsf{i}):1$ is also indistinguishable to $(t,\mathsf{i})+1$, as the random $\boldsymbol{\ell}_t[\mathsf{i},q]$ can be switched to $\mathbf{s}_{\mathrm{x}}[t,\mathsf{i}]\mathbf{s}_{\mathrm{f}}[q]$ in hybrid $(t,\mathsf{i})+1$, relying again on SXDH and the function-hiding property of IPFE. This concludes our argument of security in the simpler case where the ciphertext is queried first.

AKGS and 1-ABE for NL. Our construction of AKGS and 1-ABE essentially works for NL without modification, because the computation of a nondeterministic logspace Turing machine $M = ([Q], q_{\mathrm{acc}}, \delta)$ on an input $\mathbf{x}$ can also be represented as a sequence of matrix multiplications. We briefly describe how by pointing out the difference from L. The transition function δ of a nondeterministic TM dost not instruct a unique transition, but rather specifies a set of legitimate transitions. Following one internal configuration $(i,j,\mathbf{W},q)$, there are potentially many legitimate successors:

$$
\begin{aligned}
(i,j,\mathbf{W},q) \to \big\{ & (i'=i+\Delta i, j'=j+\Delta j, \mathbf{W}'=\mathsf{overwrite}(\mathbf{W},j,w'),q') \\
& \big|\ (q',w',\Delta i,\Delta j)\in\delta(q,\mathbf{x}[i],\mathbf{W}[j])\big\}.
\end{aligned}
$$

The computation is accepting if and only if *there exists* a path with T legitimate transitions starting from $(1,1,\mathbf{0},1)$, through $(i_t,j_t,\mathbf{W}_t,q_t)$ for $t\in[T]$, and landing at $q_T = q_{\mathrm{acc}}$.

Naturally, we modify the transition matrix as below to reflect all legitimate transitions. The only difference is that each transition block determined by δ may map a state q to multiple states q', as highlighted in the solid line box:

$$
\begin{aligned}
\mathbf{M}(\mathbf{x})[(i,j,\mathbf{W},q),(i',j',\mathbf{W}',q')] &= \mathsf{CanTransit}[(i,j,\mathbf{W}),(i',j',\mathbf{W}')] \\
&\quad \times \mathbf{M}_{\mathbf{x}[i],\mathbf{W}[j],\mathbf{W}'[j],i'-i,j'-j}[q,q'], \\
\mathbf{M}_{x,w,w',\Delta i,\Delta j}[q,q'] = 1 &\quad \text{iff } (q',w',\Delta i,\Delta j)\ \boxed{\in}\ \delta(q,x,w').
\end{aligned}
$$

Let us observe the effect of right multiplying $\mathbf{M}(\mathbf{x})$ to an $\mathbf{e}_{\mathsf{i},q}$ indicating configuration (i, q): $\mathbf{e}_{\mathsf{i},q}^{\intercal}\mathbf{M}(\mathbf{x})$ gives a vector $\mathbf{c}_1$ such that $\mathbf{c}_1[\mathsf{i}', q'] = 1$ if and only if (i', q') is a legitimate next configuration. Multiplying $\mathbf{M}(\mathbf{x})$ one more time, $\mathbf{e}_{\mathsf{i},q}^{\intercal}\big(\mathbf{M}(\mathbf{x})\big)^2$ gives $\mathbf{c}_2$ where $\mathbf{c}_2[\mathsf{i}', q']$ is the number of length-2 paths of legitimate transitions from (i, q) to (i', q'). Inductively, $\mathbf{e}_{\mathsf{i},q}^{\intercal}\big(\mathbf{M}(\mathbf{x})\big)^t$ yields $\mathbf{c}_t$ that counts the number of length-t paths from (i, q) to any other internal configuration (i', q'). Therefore, we can *arithmetize* the computation of M on $\mathbf{x}$ as

$$M|_{N,T,S}(\mathbf{x}) = \mathbf{e}_{(1,1,\mathbf{0},1)}^{\intercal}\big(\mathbf{M}(\mathbf{x})\big)^T\mathbf{t} \text{ for } \mathbf{t} = \mathbf{1}_{NS2^S} \otimes \mathbf{e}_{q_{\mathrm{acc}}}. \tag{6}$$

Right multiplying $\mathbf{t}$ in the end sums up the number of paths to $(\mathsf{i}, q_{\mathrm{acc}})$ for all i in $\mathbf{c}_T$ (i.e., accepting paths).

If the computation is not accepting — there is no path to any $(\mathsf{i}, q_{\mathrm{acc}})$ — the final sum would be 0 as desired. If the computation is accepting — there is a path to some $(\mathsf{i}, q_{\mathrm{acc}})$ — then the sum should be non-zero (up to the following technicality). Now that we have represented NL computation as matrix multiplication, we immediately obtain AKGS and 1-ABE for NL using the same construction for L.

A Technicality in the Correctness for NL. The correctness of our scheme relies on the fact that when the computation is accepting, the matrix multiplication formula (Equation (6)) counts correctly the total number of length-T accepting paths. However, a subtle issue is that in our 1-ABE, the matrix multiplications are carried out over $\mathbb{Z}_p$, where p is the order of the bilinear pairing groups. This means if the total number of accepting paths happens to be a multiple of p, the sequence of matrix multiplications mod p carried out in 1-ABE would return 0, while the correct output should be non-zero. This technicality can be circumvented if p is entropic with $\omega(\log n)$ bits of entropy and the computation $(M, \mathbf{x}, T, S)$ is independent of p. In that case, the probability that the number of accepting paths is a multiple of p is negligible. We can achieve this by letting the setup algorithm of 1-ABE sample the bilinear pairing groups from a distribution with entropic order. Then, we have statistical correctness for computations $(M, \mathbf{x}, T, S)$ chosen statically ahead of time (independent of p). We believe such *static* correctness is sufficient for most applications where correctness is meant for non-adversarial behaviors. However, if the computation $(M, \mathbf{x}, T, S)$ is chosen *adaptively* to make the number of accepting paths a multiple of p, then an accepting computation will be mistakenly rejected. We stress that security is unaffected since if an adversary chooses M and $(\mathbf{x}, T, S)$ as such, it only learns less information.

The Special Cases of DFA and NFA. DFA and NFA are special cases of L and NL, respectively, as they can be represented as Turing machines with a work tape of size $S = 1$ that always runs in time $T = N$, and the transition function δ always moves the input tape pointer to the right. Therefore, the internal configuration of a finite automaton contains only the state q, and the transition matrix $\mathbf{M}(x)$ is determined by δ and the current input bit x under scan. Different from the case of L and NL, here the transition matrix no longer

keeps track of the input tape pointer since its move is fixed — the t^{th} step uses the transition matrix $\mathbf{M}(\mathbf{x}[t])$ depending on $\mathbf{x}[t]$. Thus, the computation can be represented as follows:

$$M(\mathbf{x}) = \mathbf{e}_1^\top \prod_{t=1}^{N} \mathbf{M}(\mathbf{x}[t]) \cdot \mathbf{e}_{q_{\text{acc}}} = \mathbf{e}_1^\top \prod_{t=1}^{N} \big(\mathbf{M}_0(1-\mathbf{x}[t]) + \mathbf{M}_1\mathbf{x}[t]\big) \cdot \mathbf{e}_{q_{\text{acc}}},$$
$$\mathbf{M}_b[q,q'] = \mathbb{1}\{\delta(q,b) = q'\}.$$

Our construction of AKGS directly applies:

$$\begin{aligned} \ell_{\text{init}} &= L_{\text{init}}(\mathbf{x}) = \mathbf{e}_1^\top \mathbf{r}_0, \\ t \in [N]: \quad \boldsymbol{\ell}_t &= \big(L_{t,q}(\mathbf{x})\big)_{q\in[Q]} = -\mathbf{r}_{t-1} + \boxed{\mathbf{M}(\mathbf{x}[t])}\,\mathbf{r}_t, \qquad \big(\mathbf{r}_{t-1}, \mathbf{r}_t \xleftarrow{\$} \mathbb{Z}_p^Q\big) \\ \boldsymbol{\ell}_{N+1} &= \big(L_{N+1,q}(\mathbf{x})\big)_{q\in[Q]} = -\mathbf{r}_N + \mu \mathbf{e}_{q_{\text{acc}}}. \end{aligned}$$

When using pseudorandomness $\mathbf{r}_t[q] = \mathbf{r}_{\text{f}}[q]\mathbf{r}_{\text{x}}[t]$, the labels $\boldsymbol{\ell}_t[q]$ can be computed as the inner products of $\mathbf{v}_q = (-\mathbf{r}_{\text{f}}[q] \parallel (\mathbf{M}_0\mathbf{r}_{\text{f}})[q] \parallel (\mathbf{M}_1\mathbf{r}_{\text{f}})[q] \parallel \mathbf{0})$ and $\mathbf{u}_t = (\mathbf{r}_{\text{x}}[t-1] \parallel (1-\mathbf{x}[t])\mathbf{r}_{\text{x}}[t] \parallel \mathbf{x}[t]\mathbf{r}_{\text{x}}[t] \parallel \mathbf{0})$. Applying our 1-ABE construction with respect to such "decomposition" gives *compact* 1-ABE for DFA and NFA with secret keys of size $O(Q)$ and ciphertexts of size $O(N)$.

Discussion. Prior to our work, there have been constructions of ABE for DFA based on pairing [1,5,12,13,29,58] and ABE for NFA based on LWE [4]. However, no previous scheme achieves adaptive security unless based on q-type assumptions [12,13]. The work of [20] constructed ABE for DFA, and that of [7] for random access machines, both based on LWE, but they only support inputs of bounded length, giving up the important advantage of uniform computation of handling unbounded-length inputs. There are also constructions of ABE (and even the stronger generalization, functional encryption) for Turing machines [3,9,28,41] based on strong primitives such as multilinear map, extractable witness encryption, and indistinguishability obfuscation. However, these primitives are non-standard and currently not well-understood.

In terms of techniques, our work is most related to previous pairing-based ABE for DFA, in particular, the recent construction based on k-Lin [29]. These ABE schemes for DFA use a linear secret-sharing scheme for DFA first proposed in [58], and combining the secret key and ciphertext produces a secret-sharing in the exponent, which reveals the secret if and only if the DFA computation is accepting. Proving (even selective) security is complicated. Roughly speaking, the work of [29] relies on an *entropy propagation technique* to trace the DFA computation and propagate a few random masks "down" the computation path, with which they can argue that secret information related to states that are *backward reachable* from the final accepting states is hidden. The technique is implemented using the "nested two-slot" dual system encryption [23,33,46,47, 54,57] combined with a combinatorial mechanism for propagation.

Our AKGS is a generalization of Waters' secret-sharing scheme to L and NL, and the optimized version for DFA is identical to Waters' secret-sharing scheme. Furthermore, our 1-ABE scheme from AKGS and IPFE is more modular. In

particular, our proof (similar to our 1-ABE for non-uniform computation) does not reason about or trace the computation, and simply relies on the structure of AKGS. Using IPFE enables us to design sophisticated sequences of hybrids without getting lost in the algebra, as IPFE helps separating the logic of changes in different hybrids from how to implement the changes. For instance, we can easily manage multiple slots in the vectors encoded in IPFE for holding temporary values and generating pseudorandomness.

3 Preliminaries

Indexing. Let S be any set, we write $S^{\mathcal{I}}$ for the set of vectors whose entries are in S and indexed by $\mathcal{I}$, i.e., $S^{\mathcal{I}} = \{(\mathbf{v}[i])_{i\in\mathcal{I}} \mid \mathbf{v}[i] \in S\}$. Suppose $\mathfrak{s}_1, \mathfrak{s}_2$ are two index sets with $\mathfrak{s}_1 \subseteq \mathfrak{s}_2$. For any vector $\mathbf{v} \in \mathbb{Z}_p^{\mathfrak{s}_1}$, we write $\mathbf{u} = \mathbf{v}|^{\mathfrak{s}_2}$ for its zero-extension into $\mathbb{Z}_p^{\mathfrak{s}_2}$, i.e., $\mathbf{u} \in \mathbb{Z}_p^{\mathfrak{s}_2}$ and $\mathbf{u}[i] = \mathbf{v}[i]$ if $i \in \mathfrak{s}_1$ and 0 otherwise. Conversely, for any vector $\mathbf{v} \in \mathbb{Z}_p^{\mathfrak{s}_2}$, we write $\mathbf{u} = \mathbf{v}|_{\mathfrak{s}_1}$ for its canonical projection onto $\mathbb{Z}_p^{\mathfrak{s}_1}$, i.e., $\mathbf{u} \in \mathbb{Z}_p^{\mathfrak{s}_1}$ and $\mathbf{u}[i] = \mathbf{v}[i]$ for $i \in \mathfrak{s}_1$. Lastly, let $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_p^{\mathfrak{s}}$, denote by $\langle \mathbf{u}, \mathbf{v} \rangle$ their inner product, i.e., $\sum_{i\in\mathfrak{s}} \mathbf{u}[i]\mathbf{v}[i]$.

Coefficient Vector. We conveniently associate an affine function $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ with its coefficient vector $\mathbf{f} \in \mathbb{Z}_p^{\mathfrak{s}}$ (written as the same letter in boldface) for $\mathfrak{s} = \{\mathsf{const}\} \cup \{\mathsf{coef}_i \mid i \in \mathcal{I}\}$ such that $f(\mathbf{x}) = \mathbf{f}[\mathsf{const}] + \sum_{i\in\mathcal{I}} \mathbf{f}[\mathsf{coef}_i]\mathbf{x}[i]$.

3.1 Bilinear Pairing and Matrix Diffie-Hellman Assumption

Throughout the paper, we use a sequence of bilinear pairing groups

$$\mathcal{G} = \{(G_{\lambda,1}, G_{\lambda,2}, G_{\lambda,\mathrm{T}},\ g_{\lambda,1}, g_{\lambda,2},\ e_\lambda)\}_{\lambda\in\mathbb{N}},$$

where $G_{\lambda,1}, G_{\lambda,2}, G_{\lambda,\mathrm{T}}$ are groups of prime order $p = p(\lambda)$, and $G_{\lambda,1}$ (resp. $G_{\lambda,2}$) is generated by $g_{\lambda,1}$ (resp. $g_{\lambda,2}$). The maps $e_\lambda : G_{\lambda,1} \times G_{\lambda,_2} \to G_{\lambda,\mathrm{T}}$ are

- *bilinear:* $e_\lambda(g_{\lambda,1}^a, g_{\lambda,2}^b) = \big(e_\lambda(g_{\lambda,1}, g_{\lambda,2})\big)^{ab}$ for all a, b; and
- *non-degenerate:* $e_\lambda(g_{\lambda,1}, g_{\lambda,2})$ generates $G_{\lambda,\mathrm{T}}$.

Implicitly, we set $g_{\lambda,\mathrm{T}} = e(g_{\lambda,1}, g_{\lambda,2})$. We require the group operations as well as the bilinear maps be efficiently computable.

Bracket Notation. Fix a security parameter, for $i = 1, 2, \mathrm{T}$, we write $[\![\mathbf{A}]\!]_i$ for $g_{\lambda,i}^{\mathbf{A}}$, where the exponentiation is element-wise. When bracket notation is used, group operation is written additively, so $[\![\mathbf{A}+\mathbf{B}]\!]_i = [\![\mathbf{A}]\!]_i + [\![\mathbf{B}]\!]_i$ for matrices $\mathbf{A}, \mathbf{B}$. Pairing operation is written multiplicatively so that $[\![\mathbf{A}]\!]_1[\![\mathbf{B}]\!]_2 = [\![\mathbf{AB}]\!]_\mathrm{T}$. Furthermore, numbers can always operate with group elements, e.g., $[\![\mathbf{A}]\!]_1\mathbf{B} = [\![\mathbf{AB}]\!]_1$.

Matrix Diffie-Hellman Assumption. In this work, we rely on the MDDH assumptions defined in [26], which is implied by k-Lin.

Definition 1 (MDDH_k [26])**.** *Let $k \geq 1$ be an integer constant. For a sequence of pairing groups $\mathcal{G}$ of order $p(\lambda)$,* MDDH_k holds in G_i *($i = 1, 2, \mathrm{T}$) if*

$$\{([\![\mathbf{A}]\!]_i, [\![\mathbf{s}^\intercal\mathbf{A}]\!]_i)\}_{\lambda\in\mathbb{N}} \approx \{([\![\mathbf{A}]\!]_i, [\![\mathbf{c}^\intercal]\!]_i)\}_{\lambda\in\mathbb{N}} \text{ for } \mathbf{A} \xleftarrow{\$} \mathbb{Z}_{p(\lambda)}^{k\times(k+1)}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_{p(\lambda)}^{k}, \mathbf{c} \xleftarrow{\$} \mathbb{Z}_{p(\lambda)}^{k+1}.$$

3.2 Attribute-Based Encryption

Definition 2. *Let $\mathcal{M} = \{M_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of message sets. Let $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$ be a sequence of families of predicates, where $\mathcal{P}_\lambda = \{P : X_P \times Y_P \to \{0,1\}\}$. An* attribute-based encryption (ABE) scheme *for message space $\mathcal{M}$ and predicate space $\mathcal{P}$ consists of 4 efficient algorithms:*

- $\mathsf{Setup}(1^\lambda, P \in \mathcal{P}_\lambda)$ *generates a pair of master public/secret key* $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{KeyGen}(1^\lambda, \mathsf{msk}, y \in Y_P)$ *generates a secret key* sk_y *associated with* y.
- $\mathsf{Enc}(1^\lambda, \mathsf{mpk}, x \in X_P, g \in M_\lambda)$ *generates a ciphertext* $\mathsf{ct}_{x,g}$ *for* g *associated with* x.
- $\mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct})$ *outputs either* $\perp$ *or a message in* M_λ.

Correctness *requires that for all $\lambda \in \mathbb{N}$, all $P \in \mathcal{P}_\lambda, g \in M_\lambda$, and all $y \in Y_P$, $x \in X_P$ such that $P(x, y) = 1$,*

$$\Pr\left[\begin{array}{c}(\mathsf{mpk},\mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, P)\\ \mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, y) \;:\; \mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct}) = g\\ \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{mpk}, x, g)\end{array}\right] = 1.$$

The basic security requirement of an ABE scheme stipulates that no information about the message can be inferred as long as each individual secret key the adversary receives does not allow decryption. The adversary is given the master public key and allowed arbitrarily many secret key and ciphertexts queries. For the secret key queries, the adversary is given the secret key for a policy of its choice. For the ciphertext queries, the adversary is either given a correct encryption to the message or an encryption of a random message. It has to decide whether the encryptions it receives are correct or random. We stress that in the *adaptive* setting considered in this work, the secret key and ciphertext queries can arbitrarily interleave and depend on responses to previous queries. The definition is standard in the literature, and we refer the readers to [32] or the full version for details.

3.3 Function-Hiding Slotted Inner-Product Functional Encryption

Definition 3 (pairing-based slotted IPFE)**.** *Let $\mathcal{G}$ be a sequence of pairing groups of order $p(\lambda)$. A* slotted inner-product functional encryption (IPFE) scheme based on $\mathcal{G}$ *consists of 5 efficient algorithms:*

- $\mathsf{Setup}(1^\lambda, \mathfrak{s}_{\mathrm{pub}}, \mathfrak{s}_{\mathrm{priv}})$ *takes as input two disjoint index sets, the public slot $\mathfrak{s}_{\mathrm{pub}}$ and the private slot $\mathfrak{s}_{\mathrm{priv}}$, and outputs a pair of master public key and master secret key* $(\mathsf{mpk}, \mathsf{msk})$. *The whole index set $\mathfrak{s}$ is $\mathfrak{s}_{\mathrm{pub}} \cup \mathfrak{s}_{\mathrm{priv}}$.*
- $\mathsf{KeyGen}(1^\lambda, \mathsf{msk}, [\![\mathbf{v}]\!]_2)$ *generates a secret key $\mathsf{sk}_\mathbf{v}$ for $\mathbf{v} \in \mathbb{Z}^{\mathfrak{s}}_{p(\lambda)}$.*
- $\mathsf{Enc}(1^\lambda, \mathsf{msk}, [\![\mathbf{u}]\!]_1)$ *generates a ciphertext $\mathsf{ct}_\mathbf{u}$ for $\mathbf{u} \in \mathbb{Z}^{\mathfrak{s}}_{p(\lambda)}$ using the master* secret *key.*
- $\mathsf{Dec}(1^\lambda, \mathsf{sk}_\mathbf{v}, \mathsf{ct}_\mathbf{u})$ *is supposed to compute $[\![\langle \mathbf{u}, \mathbf{v}\rangle]\!]_{\mathrm{T}}$.*

– $\mathsf{SlotEnc}(1^\lambda, \mathsf{mpk}, [\![\mathbf{u}]\!]_1)$ *generates a ciphertext* ct *for* $\mathbf{u}|^{\mathfrak{S}}$ *when given input* $\mathbf{u} \in \mathbb{Z}_{p(\lambda)}^{\mathfrak{S}_{\mathrm{pub}}}$ *using the master* public *key.*

Decryption correctness *requires that for all* $\lambda \in \mathbb{N}$*, all index set* $\mathfrak{S}$*, and all vectors* $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_{p(\lambda)}^{\mathfrak{S}}$*,*

$$\Pr\left[\begin{array}{l}\mathsf{msk} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathfrak{S}) \\ \quad \mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, [\![\mathbf{v}]\!]_2) \ : \ \mathsf{Dec}(1^\lambda, \mathsf{sk}, \mathsf{ct}) = [\![\langle \mathbf{u}, \mathbf{v} \rangle]\!]_{\mathrm{T}} \\ \quad \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{msk}, [\![\mathbf{u}]\!]_1)\end{array}\right] = 1.$$

Slot-mode correctness *requires that for all* $\lambda \in \mathbb{N}$*, all disjoint index sets* $\mathfrak{S}_{\mathrm{pub}}, \mathfrak{S}_{\mathrm{priv}}$*, and all vector* $\mathbf{u} \in \mathbb{Z}_{p(\lambda)}^{\mathfrak{S}_{\mathrm{pub}}}$*, the following distributions should be identical:*

$$\left\{\begin{array}{r}(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathfrak{S}_{\mathrm{pub}}, \mathfrak{S}_{\mathrm{priv}}) \\ \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{msk}, [\![\mathbf{u}|^{\mathfrak{S}}]\!]_1)\end{array} : (\mathsf{mpk}, \mathsf{msk}, \mathsf{ct})\right\},$$

$$\left\{\begin{array}{r}(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathfrak{S}_{\mathrm{pub}}, \mathfrak{S}_{\mathrm{priv}}) \\ \mathsf{ct} \xleftarrow{\$} \mathsf{SlotEnc}(1^\lambda, \mathsf{mpk}, [\![\mathbf{u}]\!]_1)\end{array} : (\mathsf{mpk}, \mathsf{msk}, \mathsf{ct})\right\}.$$

Slotted IPFE generalizes both secret-key and public-key IPFEs: A secret-key IPFE can be obtained by setting $\mathfrak{S}_{\mathrm{pub}} = \varnothing$ and $\mathfrak{S}_{\mathrm{priv}} = \mathfrak{S}$; a public-key IPFE can be obtained by setting $\mathfrak{S}_{\mathrm{pub}} = \mathfrak{S}$ and $\mathfrak{S}_{\mathrm{priv}} = \varnothing$.

We now define the adaptive *function-hiding* property.

Definition 4 (function-hiding slotted IPFE)**.** *Let* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{SlotEnc})$ *be a slotted IPFE. The scheme is* function-hiding *if* $\mathsf{Exp}_{\mathrm{FH}}^0 \approx \mathsf{Exp}_{\mathrm{FH}}^1$*, where* $\mathsf{Exp}_{\mathrm{FH}}^b$ *for* $b \in \{0, 1\}$ *is defined as follows:*

– **Setup.** Run the adversary $\mathcal{A}(1^\lambda)$ and receive two disjoint index sets $\mathfrak{S}_{\mathrm{pub}}, \mathfrak{S}_{\mathrm{priv}}$ from $\mathcal{A}$. Let $\mathfrak{S} = \mathfrak{S}_{\mathrm{pub}} \cup \mathfrak{S}_{\mathrm{priv}}$. Run $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, \mathfrak{S}_{\mathrm{pub}}, \mathfrak{S}_{\mathrm{priv}})$ and return mpk to $\mathcal{A}$.
– **Challenge.** Repeat the following for arbitrarily many rounds determined by $\mathcal{A}$: In each round, $\mathcal{A}$ has 2 options.
  • $\mathcal{A}$ can submit $[\![\mathbf{v}_j^0]\!]_2, [\![\mathbf{v}_j^1]\!]_2$ for a secret key, where $\mathbf{v}_j^0, \mathbf{v}_j^1 \in \mathbb{Z}_p^{\mathfrak{S}}$. Upon this query, run $\mathsf{sk}_j \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, [\![\mathbf{v}_j^b]\!]_2)$ and return sk_j to $\mathcal{A}$.
  • $\mathcal{A}$ can submit $[\![\mathbf{u}_i^0]\!]_1, [\![\mathbf{u}_i^1]\!]_1$ for a ciphertext, where $\mathbf{u}_i^0, \mathbf{u}_i^1 \in \mathbb{Z}_p^{\mathfrak{S}}$. Upon this query, run $\mathsf{ct}_i \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{msk}, [\![\mathbf{u}_i^b]\!]_1)$ and return ct_i to $\mathcal{A}$.
– **Guess.** $\mathcal{A}$ outputs a bit b'. The outcome is b' if $\mathbf{v}_j^0|_{\mathfrak{S}_{\mathrm{pub}}} = \mathbf{v}_j^1|_{\mathfrak{S}_{\mathrm{pub}}}$ for all j and $\langle \mathbf{u}_i^0, \mathbf{v}_j^0 \rangle = \langle \mathbf{u}_i^1, \mathbf{v}_j^1 \rangle$ for all i, j. Otherwise, the outcome is 0.

Applying the techniques in [49,50] to the IPFE of [2,60], we obtain adaptively secure function-hiding slotted IPFE:

Lemma 5 ([2,49,50,60])**.** *Let* $\mathcal{G}$ *be a sequence of pairing groups and* $k \geq 1$ *an integer constant. If* MDDH_k *holds in both* G_1, G_2*, then there is an (adaptively) function-hiding slotted IPFE scheme based on* $\mathcal{G}$*.*

4 Arithmetic Key Garbling Scheme

Arithmetic key garbling scheme (AKGS) is an information-theoretic primitive related to randomized encodings [11] and partial garbling schemes [37]. It is the information-theoretic core in our construction of one-key one-ciphertext ABE (more precisely 1-ABE constructed in Section 5). Given a function $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ and two secrets $\alpha, \beta \in \mathbb{Z}_p$, an AKGS produces label functions $L_1, \ldots, L_m : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ that are affine in $\mathbf{x}$. For any $\mathbf{x}$, one can compute $\alpha f(\mathbf{x}) + \beta$ from $L_1(\mathbf{x}), \ldots, L_m(\mathbf{x})$ together with f and $\mathbf{x}$, while all other information about α, β are hidden.

Definition 6 (AKGS, adopted from Definition 1 in [37])**.** *An* arithmetic key garbling scheme (AKGS) *for a function class $\mathcal{F} = \{f\}$, where $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ for some $p, \mathcal{I}$ specified by f, consists of two efficient algorithms:*

- $\mathsf{Garble}(f \in \mathcal{F}, \alpha \in \mathbb{Z}_p, \beta \in \mathbb{Z}_p)$ *is randomized and outputs m affine functions $L_1, \ldots, L_m : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p$ (called* label functions*, which specifies how input is encoded as labels). Pragmatically, it outputs the coefficient vectors $\mathbf{L}_1, \ldots, \mathbf{L}_m$.*
- $\mathsf{Eval}(f \in \mathcal{F}, \mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}, \ell_1 \in \mathbb{Z}_p, \ldots, \ell_m \in \mathbb{Z}_p)$ *is deterministic and outputs a value in $\mathbb{Z}_p$ (the input $\ell_1, \ldots, \ell_m$ are called* labels*, which are supposed to be the values of the label functions at $\mathbf{x}$).*

Correctness *requires that for all $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p, \mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$,*

$$\Pr\left[\begin{array}{c}(\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) \\ \ell_j \leftarrow L_j(\mathbf{x}) \text{ for } j \in [m]\end{array} : \mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \alpha f(\mathbf{x}) + \beta\right] = 1.$$

We also require that the scheme have deterministic shape*, meaning that m is determined solely by f, independent of α, β, and the randomness in* Garble*. The number of label functions, m, is called the* garbling size *of f under this scheme.*

Definition 7 (linear AKGS)**.** *An AKGS* (Garble, Eval) *for $\mathcal{F}$ is* linear *if the following conditions hold:*

- $\mathsf{Garble}(f, \alpha, \beta)$ *uses a* **uniformly random** *vector $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^{m'}$ as its randomness, where m' is determined solely by f, independent of α, β.*
- *The coefficient vectors $\mathbf{L}_1, \ldots, \mathbf{L}_m$ produced by $\mathsf{Garble}(f, \alpha, \beta; \mathbf{r})$ are linear in $(\alpha, \beta, \mathbf{r})$.*
- $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m)$ *is linear in $(\ell_1, \ldots, \ell_m)$.*

Later in this paper, AKGS refers to linear AKGS by default.

The basic security notion of AKGS requires the existence of an efficient simulator that draws a sample from the real labels' distribution given $f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta$. We emphasize, as it's the same case in [37], that AKGS does *not* hide $\mathbf{x}$ and hides all other information about α, β except the value $\alpha f(\mathbf{x}) + \beta$.

Definition 8 ((usual) simulation security, Definition 1 in [37])**.** *An AKGS* (Garble, Eval) *for $\mathcal{F}$ is* secure *if there exists an efficient algorithm* Sim *such that*

for all $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p, \mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$*, the following distributions are identical:*

$$\left\{ \begin{array}{rl} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \ell_j \leftarrow L_j(\mathbf{x}) \text{ for } j \in [m] & : (\ell_1, \ldots, \ell_m) \end{array} \right\},$$
$$\{ \ (\ell_1, \ldots, \ell_m) \xleftarrow{\$} \mathsf{Sim}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta) \ : \ (\ell_1, \ldots, \ell_m) \}.$$

As discussed in Section 2.1, the usual simulation security suffices for selective (or semi-adaptive) security. To achieve adaptive security, we need the following stronger property.

Definition 9 (piecewise security)**.** *An AKGS* $(\mathsf{Garble}, \mathsf{Eval})$ *for* $\mathcal{F}$ *is* piecewise secure *if the following conditions hold:*

- *The first label is* reversely sampleable *from the other* labels *together with* f *and* $\mathbf{x}$*. This reconstruction is perfect even given all the other* label functions*. Formally, there exists an efficient algorithm* $\mathsf{RevSamp}$ *such that for all* $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p, \mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$*, the following distributions are identical:*

$$\left\{ \begin{array}{rl} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \ell_1 \leftarrow L_1(\mathbf{x}) & : (\ell_1, \mathbf{L}_2, \ldots, \mathbf{L}_m) \end{array} \right\},$$
$$\left\{ \begin{array}{rl} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \ell_j \leftarrow L_j(\mathbf{x}) \text{ for } j \in [m], j > 1 & \\ \ell_1 \xleftarrow{\$} \mathsf{RevSamp}(f, \mathbf{x}, \alpha f(\mathbf{x}) + \beta, \ell_2, \ldots, \ell_m) & : (\ell_1, \mathbf{L}_2, \ldots, \mathbf{L}_m) \end{array} \right\}.$$

- *For the other* labels*, each is* marginally random *even given all the* label functions *after it. Formally, this means for all* $f : \mathbb{Z}_p^{\mathcal{I}} \to \mathbb{Z}_p \in \mathcal{F}, \alpha, \beta \in \mathbb{Z}_p$*,* $\mathbf{x} \in \mathbb{Z}_p^{\mathcal{I}}$ *and all* $j \in [m], j > 1$*, the following distributions are identical:*

$$\left\{ \begin{array}{rl} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \ell_j \leftarrow L_j(\mathbf{x}) & : (\ell_j, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m) \end{array} \right\},$$
$$\left\{ \begin{array}{rl} (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta) & \\ \ell_j \xleftarrow{\$} \mathbb{Z}_p & : (\ell_j, \mathbf{L}_{j+1}, \ldots, \mathbf{L}_m) \end{array} \right\}.$$

As piecewise security is stronger, it implies the usual simulation security:

Lemma 10. *A* piecewise secure *AKGS for some function class is also* secure *for the same function class.*

5 1-ABE for ABPs

Arithmetic branching program (ABP) is a computation model introduced by Nisan [51] and later studied in [15,34–37]. It is defined by a directed acyclic

graph (V, E) with distinguished vertices $s, t \in V$ where every edge $e \in E$ is labeled by an affine function w_e of the input $\mathbf{x}$, and the output is computed as

$$f(\mathbf{x}) = \sum_{\substack{s\text{-}t \text{ path} \\ e_1 \cdots e_i}} \prod_{j=1}^{i} w(e_j)(\mathbf{x}).$$

Our ABE for ABPs relies on an AKGS for ABPs, which we derive as a special case of the partial garbling scheme for ABPs in [37].

Lemma 11. *There is a piecewise secure AKGS for ABPs. Moreover, the garbling size of an ABP coincides with the number of vertices in the graph.*

Below, we define and construct 1-ABE, a precursor to our full-fledged ABE, using a piecewise secure AKGS for the matching function class. It captures the key ideas for achieving adaptive security using AKGS and function-hiding IPFE, while keeping the ciphertext compact. (For technical reasons, it is more convenient to define it as a key encapsulation mechanism.)

Definition 12. *Let $\mathcal{G}$ be a sequence of pairing groups of order $p(\lambda)$. A* 1-ABE scheme based on $\mathcal{G}$ *has the same syntax as an ABE scheme in Definition 2, except that*

- *There is no message space $\mathcal{M}$.*
- Setup *outputs a master secret key* msk*, without a* mpk*.*
- $\mathsf{KeyGen}(1^\lambda, \mathsf{msk}, y, \mu)$ *outputs a secret key* sk *for policy y that encapsulates a pad $\mu \in \mathbb{Z}_{p(\lambda)}$.*
- $\mathsf{Enc}(1^\lambda, \mathsf{msk}, x)$ *uses* msk *and outputs a ciphertext* ct *for attribute x without encrypting a message.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ *outputs $\perp$ or some $[\![\mu']\!]_{\mathrm{T}}$.*
- *Correctness requires that $\mu = \mu'$ if the decapsulation should be successful, i.e., $P(x, y) = 1$.*

Such a scheme is 1-key 1-ciphertext secure *(or simply* secure*) if* $\mathsf{Exp}^0_{\text{1-sk,1-ct}} \approx \mathsf{Exp}^1_{\text{1-sk,1-ct}}$*, where* $\mathsf{Exp}^b_{\text{1-sk,1-ct}}$ *is defined as follows:*

- **Setup.** *Run the adversary $\mathcal{A}(1^\lambda)$ and receive a predicate P from it.*
- **Query I.** $\mathcal{A}$ can *submit a key query y. Upon this query, sample two random pads $\mu^0, \mu^1 \xleftarrow{\$} \mathbb{Z}_{p(\lambda)}$, run $\mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda, \mathsf{msk}, y, \mu^0)$, and return (sk, μ^b) to $\mathcal{A}$.*
- **Challenge.** *$\mathcal{A}$ submits a challenge attribute x. Upon the challenge, run $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(1^\lambda, \mathsf{msk}, x)$, and return ct to $\mathcal{A}$.*
- **Query II.** *Same as Query I.*
- **Guess.** *$\mathcal{A}$ outputs a bit b'. The outcome of the experiment is b' if the adversary makes only a single key query for some y and $P(x, y) = 0$. Otherwise, the outcome is 0.*

For any function class $\mathcal{F}$ (e.g., arithmetic branching programs), we show how to construct a 1-ABE for the class of zero-test predicates in $\mathcal{F}$ (i.e., predicates of form $f_{\neq 0}, f_{=0}$ that computes whether $f(\mathbf{x})$ evaluates to zero or non-zero), using a piecewise secure AKGS for $\mathcal{F}$ and a function-hiding secret-key IPFE scheme.

Construction 13 (1-ABE)**.** We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ below for simplicity of notations. Let $(\mathsf{Garble}, \mathsf{Eval})$ be an AKGS for a function class $\mathcal{F}$, $\mathcal{G}$ pairing groups of order p, and $(\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec})$ a secret-key IPFE based on $\mathcal{G}$. We construct a 1-ABE scheme based on $\mathcal{G}$ for the predicate space $\mathcal{P}$ induced by $\mathcal{F}$:

$$X_n = \mathbb{Z}_p^n, \quad Y_n = \{f_{\neq 0}, f_{=0} \mid f \in \mathcal{F}, f : \mathbb{Z}_p^n \to \mathbb{Z}_p\},$$
$$\mathcal{P} = \{P_n : X_n \times Y_n \to \{0,1\}, (\mathbf{x}, y) \mapsto y(\mathbf{x}) \mid n \in \mathbb{N}\}.$$

The 1-ABE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ operates as follows:

- $\mathsf{Setup}(1^n)$ takes the attribute length in unary (i.e., P_n is encoded as 1^n) as input. It generates an IPFE master secret key $\mathsf{msk} \xleftarrow{\$} \mathsf{IPFE.Setup}(\mathfrak{s}_{\text{1-ABE}})$ for the index set $\mathfrak{s}_{\text{1-ABE}} = \{\mathsf{const}, \mathsf{coef}_1, \ldots, \mathsf{coef}_n, \mathsf{sim}_1, \mathsf{sim}_\star\}$. The algorithm returns msk as the master secret key.

  Note: *The positions indexed by* $\mathsf{const}, \mathsf{coef}_1, \ldots, \mathsf{coef}_n$ *in the secret key encode the coefficient vectors* $\mathbf{L}_j$ *of the label functions* L_i *produced by garbling* f *with secrets* α, β*, and these positions encode* $(1, \mathbf{x})$ *in the ciphertext. The positions indexed by* $\mathsf{sim}_1, \mathsf{sim}_\star$ *are set to zero by the honest algorithms, and are only used in the security proof.*
- $\mathsf{KeyGen}(\mathsf{msk}, y \in Y_n, \mu \in \mathbb{Z}_p)$ samples $\eta \xleftarrow{\$} \mathbb{Z}_p$ and garbles the function f underlying y as follows:

$$\begin{cases} \alpha \leftarrow \mu, \ \ \beta \leftarrow 0, & \text{if } y = f_{\neq 0}; \\ \alpha \leftarrow \eta, \ \ \beta \leftarrow \mu, & \text{if } y = f_{=0}; \end{cases} \qquad (\mathbf{L}_1, \ldots, \mathbf{L}_m) \xleftarrow{\$} \mathsf{Garble}(f, \alpha, \beta).$$

  It generates an IPFE key $\mathsf{isk}_j \xleftarrow{\$} \mathsf{IPFE.KeyGen}(\mathsf{msk}, [\![\mathbf{v}_j]\!]_2)$ for the following vector $\mathbf{v}_j$ encoding each label function L_j:

vector	const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathbf{v}_j$	$\mathbf{L}_j[\mathsf{const}]$	$\mathbf{L}_j[\mathsf{coef}_i]$	0	0

  The algorithm returns $\mathsf{sk}_y = (y, \mathsf{isk}_1, \ldots, \mathsf{isk}_m)$ as the secret key.
- $\mathsf{Enc}(\mathsf{msk}, \mathbf{x} \in \mathbb{Z}_p^n)$ generates an IPFE ciphertext $\mathsf{ict} \xleftarrow{\$} \mathsf{IPFE.Enc}(\mathsf{msk}, [\![\mathbf{u}]\!]_1)$ encrypting the vector $\mathbf{u}$ that contains $1, \mathbf{x}$:

vector	const	coef_i	sim_1	$\mathsf{sim}_\star$
$\mathbf{u}$	1	$\mathbf{x}[i]$	0	0

  It returns $\mathsf{ct} = (\mathbf{x}, \mathsf{ict})$ as the ciphertext.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ parses sk as $(y, \mathsf{isk}_1, \ldots, \mathsf{isk}_m)$ and ct as $(\mathbf{x}, \mathsf{ict})$, and returns $\perp$ if $y(\mathbf{x}) = 0$. Otherwise, it does the following:

$$\text{for } j \in [m]: \quad [\![\ell_j]\!]_{\mathrm{T}} \leftarrow \mathsf{IPFE.Dec}(\mathsf{isk}_j, \mathsf{ict}),$$
$$[\![\mu']\!]_{\mathrm{T}} \leftarrow \begin{cases} \frac{1}{f(\mathbf{x})}\mathsf{Eval}(f, \mathbf{x}, [\![\ell_1]\!]_{\mathrm{T}}, \ldots, [\![\ell_m]\!]_{\mathrm{T}}), & \text{if } y = f_{\neq 0}; \\ \mathsf{Eval}(f, \mathbf{x}, [\![\ell_1]\!]_{\mathrm{T}}, \ldots, [\![\ell_m]\!]_{\mathrm{T}}), & \text{if } y = f_{=0}. \end{cases}$$

The algorithm returns $[\![\mu']\!]_{\mathrm{T}}$ as the decapsulated pad.

Note: *We show the correctness of the scheme. First, by the correctness of IPFE and the definition of vectors* $\mathbf{v}_j, \mathbf{u}$*, we have* $\ell_j = \langle \mathbf{u}, \mathbf{v}_j \rangle = L_j(\mathbf{x})$ *for all* $j \in [m]$*. Next, by the linearity of* Eval *in* $\ell_1, \ldots, \ell_m$*, we can evaluate the garbling in the exponent of the target group and obtain* $\mathsf{Eval}(f, \mathbf{x}, \ell_1, \ldots, \ell_m) = \alpha f(\mathbf{x}) + \beta$ *in the exponent. In the two cases where decapsulation should succeed, we have*

$$\alpha f(\mathbf{x}) + \beta = \begin{cases} \mu f(\mathbf{x}), & \text{if } y = f_{\neq 0} \text{ and } f(\mathbf{x}) \neq 0; \\ \mu, & \text{if } y = f_{=0} \text{ and } f(\mathbf{x}) = 0. \end{cases}$$

In both cases, the μ' *above equals to* μ*. Therefore,* Dec *correctly decapsulates the pad.*

Theorem 14. *Suppose in Construction 13, the AKGS is* piecewise secure *and the IPFE scheme is* function-hiding*, then the constructed 1-ABE scheme is* 1-key 1-ciphertext secure*.*

We refer the readers to the full version for the formal proof.

Acknowledgments. The authors were supported by NSF grants[12] CNS-1528178, CNS-1929901, CNS-1936825 (CAREER). The authors thank Hoeteck Wee for helpful discussions and the anonymous reviewers for insightful comments.

References

1. Shashank Agrawal and Melissa Chase. Simplifying design and analysis of complex predicate encryption schemes. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 627–656. Springer, Heidelberg, April / May 2017.
2. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016.
3. Shweta Agrawal and Monosij Maitra. FE and iO for turing machines from minimal assumptions. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 473–512. Springer, Heidelberg, November 2018.
4. Shweta Agrawal, Monosij Maitra, and Shota Yamada. Attribute based encryption (and more) for nondeterministic finite automata from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 765–797. Springer, Heidelberg, August 2019.
5. Shweta Agrawal, Monosij Maitra, and Shota Yamada. Attribute based encryption for deterministic finite automata from DLIN. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 91–117. Springer, Heidelberg, December 2019.

[12] The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

6. Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. Delegating RAM computations with adaptive soundness and privacy. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 3–30. Springer, Heidelberg, October / November 2016.
7. Prabhanjan Ananth, Xiong Fan, and Elaine Shi. Towards attribute-based encryption for RAMs from LWE: Sub-linear decryption, and more. Cryptology ePrint Archive, Report 2018/273, 2018. https://eprint.iacr.org/2018/273.
8. Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 152–181. Springer, Heidelberg, April / May 2017.
9. Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for turing machines. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 125–153. Springer, Heidelberg, January 2016.
10. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175. IEEE Computer Society Press, October 2004.
11. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 120–129. IEEE Computer Society Press, October 2011.
12. Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 557–577. Springer, Heidelberg, May 2014.
13. Nuttapong Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 591–623. Springer, Heidelberg, December 2016.
14. Nuttapong Attrapadung. Dual system framework in multilinear settings and applications to fully secure (compact) ABE for unbounded-size circuits. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 3–35. Springer, Heidelberg, March 2017.
15. Amos Beimel and Anna Gal. On arithmetic branching programs. In *IN PROC. OF THE 13TH ANNUAL IEEE CONFERENCE ON COMPUTATIONAL COMPLEXITY*, pages 68–80, 1998.
16. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 321–334, May 2007.
17. Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 470–491. Springer, Heidelberg, November / December 2015.
18. Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 439–448. ACM Press, June 2015.
19. Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.

20. Xavier Boyen and Qinyi Li. Attribute-based encryption for finite automata from LWE. In Man Ho Au and Atsuko Miyaji, editors, *ProvSec 2015*, volume 9451 of *LNCS*, pages 247–267. Springer, Heidelberg, November 2015.
21. Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 429–437. ACM Press, June 2015.
22. Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, Heidelberg, April 2015.
23. Jie Chen, Junqing Gong, Lucas Kowalczyk, and Hoeteck Wee. Unbounded ABE via bilinear entropy expansion, revisited. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 503–534. Springer, Heidelberg, April / May 2018.
24. Jie Chen, Junqing Gong, and Hoeteck Wee. Improved inner-product encryption with adaptive security and full attribute-hiding. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 673–702. Springer, Heidelberg, December 2018.
25. Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 164–195. Springer, Heidelberg, March 2016.
26. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.
27. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2013.
28. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013.
29. Junqing Gong, Brent Waters, and Hoeteck Wee. ABE for DFA from k-lin. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 732–764. Springer, Heidelberg, August 2019.
30. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013.
31. Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient ABE for branching programs. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 550–574. Springer, Heidelberg, November / December 2015.
32. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

33. Dennis Hofheinz, Jessica Koch, and Christoph Striecks. Identity-based encryption with (almost) tight security in the multi-instance, multi-ciphertext setting. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 799–822. Springer, Heidelberg, March / April 2015.
34. Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *In Proc. of 5th ISTCS*, pages 174–183, 1997.
35. Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st FOCS*, pages 294–304. IEEE Computer Society Press, November 2000.
36. Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *ICALP 2002*, volume 2380 of *LNCS*, pages 244–256. Springer, Heidelberg, July 2002.
37. Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 650–662. Springer, Heidelberg, July 2014.
38. Zahra Jafargholi, Chethan Kamath, Karen Klein, Ilan Komargodski, Krzysztof Pietrzak, and Daniel Wichs. Be adaptive, avoid overcommitting. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 133–163. Springer, Heidelberg, August 2017.
39. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *Journal of Cryptology*, 26(2):191–224, April 2013.
40. Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. Function-hiding inner product encryption is practical. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 544–562. Springer, Heidelberg, September 2018.
41. Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 521–551. Springer, Heidelberg, August 2019.
42. Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 419–428. ACM Press, June 2015.
43. Lucas Kowalczyk, Jiahui Liu, Tal Malkin, and Kailash Meiyappan. Mitigating the one-use restriction in attribute-based encryption. In Kwangsu Lee, editor, *ICISC 18*, volume 11396 of *LNCS*, pages 23–36. Springer, Heidelberg, November 2019.
44. Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for NC^1 from k-Lin. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 3–33. Springer, Heidelberg, May 2019.
45. Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May / June 2010.

46. Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479. Springer, Heidelberg, February 2010.
47. Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, Heidelberg, May 2011.
48. Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 180–198. Springer, Heidelberg, August 2012.
49. Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Heidelberg, August 2017.
50. Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th FOCS*, pages 11–20. IEEE Computer Society Press, October 2016.
51. Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *23rd ACM STOC*, pages 410–418. ACM Press, May 1991.
52. Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 214–231. Springer, Heidelberg, December 2009.
53. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, August 2010.
54. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 349–366. Springer, Heidelberg, December 2012.
55. Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 457–473. Springer, Heidelberg, March 2009.
56. Junichi Tomida, Masayuki Abe, and Tatsuaki Okamoto. Efficient functional encryption for inner-product values with full-hiding security. In Matt Bishop and Anderson C. A. Nascimento, editors, *ISC 2016*, volume 9866 of *LNCS*, pages 408–425. Springer, Heidelberg, September 2016.
57. Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009.
58. Brent Waters. Functional encryption for regular languages. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 218–235. Springer, Heidelberg, August 2012.
59. Hoeteck Wee. Dual system encryption via predicate encodings. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 616–637. Springer, Heidelberg, February 2014.
60. Hoeteck Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 206–233. Springer, Heidelberg, November 2017.
61. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.