FAME: Fast Attribute-based Message Encryption

Shashank Agrawal*
Visa Research
shaagraw@visa.com

Melissa Chase
Microsoft Research
melissac@microsoft.com

ABSTRACT

Time and again, attribute-based encryption has been shown to be the natural cryptographic tool for building various types of conditional access systems with far-reaching applications, but the deployment of such systems has been very slow. A central issue is the lack of an encryption scheme that can operate on sensitive data very efficiently and, at the same time, provides features that are important in practice.

This paper proposes the first fully secure ciphertext-policy and key-policy ABE schemes based on a standard assumption on Type-III pairing groups, which do not put any restriction on policy type or attributes. We implement our schemes along with several other prominent ones using the Charm library, and demonstrate that they perform better on almost all parameters of interest.

1 INTRODUCTION

Over the course of a decade, attribute-based encryption (ABE) [52] has been shown to have applications in a variety of settings like network privacy [12], pay-per-view broadcasting [55], health record access-control [9, 18], cloud security [53], verifiable computation [49], and forward-secure messaging [32]. Moreover, companies like Zeutro [1] use ABE to provide data security solutions for cloud applications. This should not come as a surprise: as opposed to the all-or-nothing approach of public-key encryption, ABE provides a much more fine-grained control of encrypted data.

In a ciphertext-policy ABE (CP-ABE) scheme [28], for instance, ciphertexts are attached to access policies and keys are associated with sets of attributes. A key is able to recover the message hidden in a ciphertext if and only if the set of attributes *satisfy* the access policy. To give an example, a policy P could say '(Zipcode:90210 OR City:BeverlyHills) AND (AgeGroup:18-25)' and an individual A could have a key for {Zipcode:90210, AgeGroup:Over65}, in which case A would not be able to decrypt any message encrypted under P. A key policy (KP-ABE) scheme, on the other hand, is the *dual* of CP-ABE with ciphertexts attached to attribute sets and keys associated with access policies.

Despite being such a versatile cryptographic tool, ABE's impact on the real world has been limited. A central issue is the lack of schemes that not only have strong security guarantees and fast operations, but provide features that are highly desirable in practice. In this paper, we propose new ABE schemes that simultaneously:

(1) put no restriction on size of policies or attribute sets;
(2) allow any arbitrary string to be used as an attribute;
(3) are based on the faster Type-III pairing groups;
(4) need a small number of pairings for decryption; and,
(5) satisfy the natural security requirement under a standard hardness assumption. [1]

Each of these properties are crucial to make an ABE scheme usable in the real world. As far as we know, our schemes are the first to achieve all of them.

Furthermore, our schemes' performance compares quite favorably with the most prominent and state-of-the-art schemes in literature. Consider for example the CP-ABE scheme of Bethencourt, Sahai, and Waters [16] (BSW), which is arguably the most popular ABE scheme among application designers, mainly due to its simple structure and remarkable efficiency. However, security of this scheme is not known to follow from a standard cryptographic assumption. Our new CP-ABE scheme not only gives full security under a standard assumption, but also encrypts, decrypts, and generates keys faster than BSW. In particular, decryption time is a mere 0.06s even if as many as 100 attributes are involved, whereas BSW takes more than 2s. Our ciphertexts and keys are 25% smaller too. Thus we believe that our more secure scheme can replace BSW as the de facto instantiation of the ABE component in most applications (policy-sealed data [53] is one example), while substantially improving the application's performance at the same time.

We now argue why the properties our schemes satisfy are important to build a fast and usable ABE scheme.

Policies & attributes. As institutions grow, more and more complex roles, entities, policies, procedures, etc. are added on a regular basis. However, most ABE schemes known in literature put one or the other restriction on what can be encoded into ciphertexts and keys. These restrictions are in the form of bounds that need to be fixed before an ABE system is deployed. For example, there could be a bound on the number of attributes that could be encoded into a key/ciphertext [4, 40, 43] or the size of access policies [19, 27, 57].

Such bounds not only limit the expressiveness of an ABE scheme, but also adversely affect the time and space complexity of various operations. A generous bound can slow down an ABE system considerably, while a tight bound can only serve well for a limited amount of time (after which a new system with a larger bound would have to be deployed, requiring all data to be re-encrypted and new keys to be generated). Our ABE schemes, on the other hand, do not put any restriction on the size of policies or attribute sets that can be encoded.

Attribute usage. Recall the policy P we defined earlier, given by '(Zipcode:90210 OR City:BeverlyHills) AND (AgeGroup:18-25)'. Suppose an ABE system encrypts some secret data under this policy. If the user base is spread across the United States, then the system

*Part of this work was done when the author was an intern at Microsoft Research, Redmond.

[1] To prove security, we model the hash function in our constructions as a random oracle (RO). Note that all ABE schemes in literature that support an unlimited number of attributes from an unbounded set (like we do) are proven secure in the RO model. Moreover, the use of RO is fairly common in many cryptographic protocols used in practice like Full Domain Hash signatures [14] and OAEP encryption [15].

should be able to issue keys for every zip code and city. Many prominent ABE schemes in literature are small universe: they require an a-priori polynomial bound on the number of different attributes that could ever be used [19, 36, 42, 57, 58]. The size of public-key then scales linearly with this bound (the set-up time is affected similarly). We have roughly 43000 different zip codes in the US and about 20000 different cities; if there are group elements for each one of them then the public key will become very large.

But the problem does not stop here. What if we like to put user names, addresses, etc. as part of the policies? The number of different attributes will not just be in hundreds of millions, they will grow rapidly with time. Though there are large universe ABE constructions too [5, 10, 44, 50], they are not ideal because of the necessity to map attributes to group elements manually. Our schemes, in contrast, allow *any* string to be used as an attribute, be it names of people, home addresses, etc. The mapping is via a hash function which is modeled as a random oracle in the security proof.

Pairings. As of today, pairing friendly elliptic curves are the only mathematical structures available for building practical ABE schemes. They are given by a triple of groups $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ with an efficiently computable map e that associates pairs of elements from groups $\mathbb{G}$ and $\mathbb{H}$ with elements in $\mathbb{G}_T$. Among them, asymmetric prime-order (Type-III) pairings have been the recommended choice by experts [25]. The other two options are not suitable: composite-order curves have large representation[2] and very slow pairing operations [33]; and symmetric prime-order (Type-I) groups have serious security issues [24, 37]. Note that in constrast to ABE schemes that are proven secure under Type-I setting and then implemented in Type-III [51], we use only Type-III throughout the paper.

Decryption. The decryption procedure in an ABE scheme is arguably the most important one. It is this function that the users of an ABE system invoke most of the times, often on computationally weak devices. In much of the initial work in ABE [28, 42, 57], decryption was fairly expensive, particularly for complex access structures, because a pairing computation was needed for *each* attribute. Our schemes require only 6 pairing operations for decryption, regardless of the number of attributes involved! This leads to significant savings in decryption time. Furthermore, our encryption and key generation procedures operate primarily in the smaller and faster source group of a bilinear map. Concretely, any ciphertext or key has 3 elements from this group per attribute, and only a constant number (just 3) from the other. This unique feature of our schemes makes it even more practical.

Security. Our schemes satisfy the natural security requirement for ABE, which is commonly known as full or adaptive security. A fully secure scheme provides confidentiality for data encrypted under policies chosen *anytime* during a system's life-cycle, even after the system parameters have been published and several keys have been distributed. On the contrary, *selectively* secure schemes can only guarantee security for policies that are declared upfront, i.e. before the system is deployed [27, 28, 48].

[2]Since the elliptic curve group order must be infeasible to factor, it must be at least (say) 1024 bits. On the other hand, a 160-bit prime-order elliptic curve group provides an equivalent level of security (NIST SP 800-57).

Further, our ABE schemes are proven secure under (a variant of) the decisional linear assumption (DLIN) [17], which is a well-understood and time-tested hardness assumption on bilinear pairings. This gives a high level of confidence in security as opposed to q-type assumptions [5, 11, 50, 57] which are fairly new and not very well understood. Moreover, many different types of q-type assumptions have been proposed in the context of ABE, and it is not clear how they compare with each other or with the standard assumptions.

See Table 1.1 for a property-wise comparison of our schemes against the most prominent and state-of-the-art schemes in literature.

Predicate encryption. The starting point for the design of our ABE schemes is the recent work of Chen, Gay, and Wee [19]. They give encryption schemes not just for ABE but a variety of other predicates like inner-product [38], building on the predicate encoding [58] and dual system group [21] abstractions. However, their ABE schemes are small universe (need an a-priori bound on the number of attributes) and put restrictions on the policies that can be used. We show how to overcome these problems *without compromising performance* with the help of new techniques in the following section. In fact, our schemes perform better than Chen et al. on almost all metrics of interest. We believe that our ideas can also be used to improve the efficiency of non-ABE type problems studied in their work.

Implementation. We implement our CP-ABE and KP-ABE schemes in the Charm framework [7] along with the most prominent and state-of-the-art schemes in literature. We rigorously compare their performance on various parameters and test cases. Our analysis highlights the trade-offs between newer schemes like our own and Chen et al. [19] (for which no implementation was available), and older ones like Bethencourt et al. [16] and Goyal et al. [28]. In particular, our CP-ABE achieves faster encryption and key generation times than any previous fully secure scheme—even faster than Bethencourt et al. which is secure only in the generic group model but has been used in a number of implementations. It also has significantly faster decryption times than all of the selectively secure schemes. See Figures 5.1 and 5.2 in §5 for the performance of the algorithms of each scheme under various test cases.

Concretely, our CP-ABE scheme always takes only 0.10s to be set-up, 0.24s to generate a key for 10 attributes, and 0.16s to encrypt data under a policy that requires all 10 attributes for decryption, *on an ordinary laptop.* More importantly, the time required for decryption is a mere 0.06s even if as many as 100 attributes are involved because we always use only 6 pairing operations. In contrast, number of pairings required by Bethencourt et al. and (the fastest version of) Waters [57] scales linearly with the number of attributes. Their decryption time is more than 1s and 2s for 100 attributes, respectively.

See Table 1.2 for a *qualitative* comparison of various ABE schemes in terms of the running time of different algorithms.

We also analyze why one scheme performs better than the other by breaking down the algorithms of the schemes into the number of different types of group operations they need and looking at the amount of time each one of them takes. This provides a very

Scheme	*Unrestricted policies*	*Arbitrary attributes*	*Type-III pairings*	*Full security*	*Standard assumption*
Our CP-ABE (Fame)	✓	✓	✓	✓	✓
Chen et al. [20, Appendix B.2]	✓	✗	✓	✓	✓
Waters [57, Section 3]	✓	✗	✗	✗	✗
Bethencourt et al. [16, Section 4.2]	✓	✓	✗	✓	✗
Our KP-ABE	✓	✓	✓	✓	✓
Chen et al. [20, Appendix B.1]	✗	✗	✓	✓	✓
Goyal et al. [29, Appendix A.1]	✓	✗	✗	✗	✓

Table 1.1: A property-wise comparison of the various ABE schemes we consider. The upper and lower parts of the table list the CP-ABE and KP-ABE schemes respectively. Please see the relevant parts of the introduction for a discussion of why each of the properties are important.

Scheme	*Set-up*	*Key generation*	*Encryption*	*Decryption*
Our CP-ABE (Fame)	●●●	●●	●●◖	●●●
Chen et al. [20, Appendix B.2] (SXDH)	●	●◖	●●	●●●
Chen et al. [20, Appendix B.2] (DLIN)	●	●	●	●●●
Waters [57, Section 3]	●●	●●●	●●	●●
Bethencourt et al. [16, Section 4.2]	●●●	●●	●●	●
Our KP-ABE	●●●	●●	●	●●●
Chen et al. [20, Appendix B.1] (SXDH)	●	●●	●●◖	●●●
Chen et al. [20, Appendix B.1] (DLIN)	●	●	●	●●●
Goyal et al. [29, Appendix A.1]	●●	●●	●●◖	●

Table 1.2: A qualitative comparison of the various ABE schemes we consider in terms of the running time of different algorithms. More the number of circles, the better the efficiency (lower running time). The upper and lower parts of the table list the CP-ABE and KP-ABE schemes respectively. Please see §5 for a concrete and thorough analysis. Note that we have implemented two versions of Chen et al.'s ABE schemes, one secure under the symmetric Diffie-Hellman assumption (SXDH) and the other under the decisional linear (DLIN) assumption. Our schemes are secure under (a variant of) the latter assumption.

fine-grained view of how the various schemes compare with each other. See Tables 5.1, 5.3, 5.2 and 5.4 in §5.

Lastly, our schemes have shorter ciphertexts and keys than most of the schemes compared with. There is 25% savings in ciphertext- and key-size w.r.t. Bethencourt et al. and 50% savings in key-size w.r.t. the fastest version of Chen et al. (Table 5.5).

The implementation code is available on GitHub [2].

Organization. Our primary focus will be on designing and analyzing a CP-ABE scheme called Fame because, traditionally, it has been harder to build than KP-ABE[3] and seems to have more practical applications. In the remainder of this section we discuss the intuition behind this construction. In §2 we describe our notation and define attribute-based encryption formally. In §3 we present Fame in full detail and then, in §4, we prove its security under the decisional linear assumption. We analyze the performance of Fame vis-à-vis several other prominent CP-ABE schemes in §5. Some more related work is surveyed in §6.

We provide a formal description of our KP-ABE scheme in Appendix B but skip a proof of security since it is similar to that of Fame. In §5 we briefly discuss the performance of this scheme with respect to two other schemes we implemented.

[3]The first proposal of KP-ABE in 2006 [28] was already under a standard assumption, but until the work of Waters in 2011 [57], there was no such scheme for CP-ABE. In an earlier paper [27], a generic method for converting KP-ABE to CP-ABE was proposed but it leads to a significant blow-up in encryption and decryption time.

1.1 Designing our ABE schemes

Monotone span programs. In order to study the type of access policies used in practice, Boolean formulas provide a very good representation. However, a more general class called monotone span programs (MSPs) is better suited to the design of encryption schemes. Indeed, barring a few original proposals for ABE [16, 28, 48], the majority of later work has used MSPs. (A Boolean formula with AND and OR gates can be easily converted into an MSP—see §2 for a formal discussion).

An MSP is given by a matrix $\mathbf{M}$ and a function π that maps each row of $\mathbf{M}$ to an attribute. $(\mathbf{M}, \pi)$ also act as a linear secret-sharing scheme. A secret value can be split into shares via $\mathbf{M}$, with one share for every row. If a set of attributes S *satisfies* $(\mathbf{M}, \pi)$, then one can linearly combine the shares of the rows mapping to attributes in S to recover the secret.

High-level design of CP-ABEs. At a high level, a CP-ABE scheme supporting MSPs works as follows. A key has some component sk_y for each attribute y in S, which generally consists of one or more elements from a group $\mathbb{H}$. These components must be tied together properly in order to prevent parties from combining two or more keys to decrypt a ciphertext that none of them is individually supposed to. Likewise, a ciphertext has a component ct_i made up of elements from a group $\mathbb{G}$ for the ith row of $\mathbf{M}$. This component masks the ith row's share with *some special value*, which must be present in the $\mathsf{sk}_{\pi(i)}$ component of the key *in some form*, so that a user with attribute $\pi(i)$ is able to recover the ith share during decryption. The public parameters generated during system set-up provide such values for ciphertexts and keys. Intuitively, we need some unique group elements for each attribute in the system, otherwise a single key component may be able to reveal multiple shares in a ciphertext.

CGW scheme. The recent work of Chen, Gay, and Wee [19], referred to as CGW hereafter, builds compact ABE schemes using Type-III pairings. Their first step is to pick matrices $\mathbf{A}$ and $\mathbf{B}$ over integers modulo a prime which embed the k-linear assumption [54]. Suppose $\mathbf{a}^{\perp}$ and $\mathbf{b}^{\perp}$ are vectors orthogonal to $\mathbf{A}$ and $\mathbf{B}$ respectively. A simple basis given by $([\mathbf{A}]_1, [\mathbf{b}^{\perp}]_1)$ and $([\mathbf{B}]_2, [\mathbf{a}^{\perp}]_2)$ is chosen for ciphertexts and keys respectively, where the subscript 1, for instance, denotes a mapping to group $\mathbb{G}$. Then, for each attribute x in the universe, they define a *new* pair of bases $([\mathbf{W}_x^{\top}\mathbf{A}]_1, [\mathbf{W}_x^{\top}\mathbf{b}^{\perp}]_1)$ and $([\mathbf{W}_x\mathbf{B}]_2, [\mathbf{W}_x\mathbf{a}^{\perp}]_2)$ by choosing a random matrix $\mathbf{W}_x$. If *matching* components of a ciphertext and key are paired, i.e., those generated w.r.t. $[\mathbf{W}_x^{\top}\mathbf{A}]_1$, $[\mathbf{A}]_1$ and $[\mathbf{W}_x\mathbf{B}]_2$, $[\mathbf{B}]_2$ respectively, then observe that this leads to *cancellation* in the sense that

$$(\mathbf{W}_x^{\top}\mathbf{A})^{\top}\mathbf{B} = \mathbf{A}^{\top}(\mathbf{W}_x\mathbf{B}), \tag{1.1}$$

but pairing with $[\mathbf{W}_y\mathbf{B}]_2$, $[\mathbf{B}]_2$ for $y \neq x$ does not. CGW calls this the *associativity* property.

Challenges. While CGW's work advances the state-of-the-art for ABE, it has some notable drawbacks. First, their schemes are small universe: one needs to know the total number ℓ of different attributes that will ever be needed in advance, so that the matrices $[\mathbf{W}_1^{\top}\mathbf{A}]_1, \ldots, [\mathbf{W}_\ell^{\top}\mathbf{A}]_1$ can be placed inside the public-key. Second, their KP-ABE scheme can only support MSPs with an a-priori bounded number of columns, which roughly translates to Boolean formulas with a limited number of AND gates. Set-up time and size of parameters both scale linearly with this bound (and with ℓ).

How do we support arbitrary attributes – any number of them, and allow any access policy to be used *without* blowing up the size of public parameters out of proportion? Let us focus on the former problem for now. A simple idea that comes to mind is to use a hash function $\mathcal{H}$ to generate $[\mathbf{W}_x^{\top}\mathbf{A}]_1$ in ciphertexts and $[\mathbf{W}_x\mathbf{B}]_2$ in keys for an attribute x. There are several problems with this approach:

- $\mathbb{G}$ and $\mathbb{H}$ have a very different structure since we are in the Type-III setting [25]. Hashing any string into them would produce completely unrelated values.
- The discrete logs of the hashed values should not be revealed, otherwise it would not be possible to argue security.[4]
- Suppose $[\mathbf{W}_x^{\top}\mathbf{A}]_1$ is generated through $\mathcal{H}$ during the encryption process. How can the key issuer generate $[\mathbf{W}_x\mathbf{B}]_2$ without explicit knowledge of $\mathbf{W}_x$?

Such types of problems arise in many other schemes too. Take for instance the small universe KP-ABE scheme of Goyal et al. [28]. It uses g^{t_x} in the ciphertext and g^{1/t_x} in the key for an attribute x, where g^{t_x} is provided as part of the public key. Without knowledge of t_x, g^{1/t_x} cannot be generated, so the master secret key must contain it. But what if g^{t_x} is derived directly from a hash function, so that t_x is not available at all? As another example, the schemes of Okamoto and Takashima [46, 47] use a vector of group elements for each attribute to form a ciphertext and an orthogonal vector to form the key. If the former vector is generated through a hash function, it is completely unclear how to generate the latter to use in the key.

Note that both Goyal et al.'s and Okamoto and Takashima's schemes are built upon symmetric groups, whereas CGW's schemes are designed in the asymmetric setting, which only makes solving the problems discussed above harder.

Approach. Associativity property (1.1) can help us find a way around the issue of asymmetry. Observe that a basis of type $[\mathbf{W}_x^{\top}\mathbf{A}]_1$ is not paired with $[\mathbf{W}_y\mathbf{B}]_2$ for any y. Thus it is conceivable to have them in the same group, while keeping $\mathbf{A}, \mathbf{B}$ (with which $\mathbf{W}_x^{\top}\mathbf{A}$, $\mathbf{W}_x\mathbf{B}$ are actually paired) in the other.

Even if $\mathbf{W}_x^{\top}\mathbf{A}$, $\mathbf{W}_x\mathbf{B}$ are in the same group, we still need to find a way to generate them through $\mathcal{H}$. Suppose one can generate $[\mathbf{W}_x^{\top}\mathbf{A}]_1$ with the help of $\mathcal{H}$ somehow, how would she produce $[\mathbf{W}_x\mathbf{B}]_1$ without explicit knowledge of $\mathbf{W}_x$? We take a different approach here: we discover a way to generate keys with the help of $[\mathbf{W}_x^{\top}\mathbf{A}]_1$ and $\mathbf{B}$ only! As a result, the structure of our keys is very different from that of CGW. While their keys are in the basis $[\mathbf{W}_x\mathbf{B}]_2$, our keys end up having an additional random component in the direction of $\mathbf{a}^{\perp}$, the vector orthogonal to $\mathbf{A}$. Removing this extra *noise* necessitates a more sophisticated analysis than CGW. Indeed, we use an extra layer of *hybrids* on top of theirs to get rid of the extra component.

FAME's ciphertexts and keys have elements from both groups $\mathbb{G}$ and $\mathbb{H}$ because, recall that, $\mathbf{W}_x^{\top}\mathbf{A}$ and $\mathbf{A}$ as well as $\mathbf{W}_y\mathbf{B}$ and $\mathbf{B}$ reside in different groups. Thus we do not know how to prove security of FAME from the symmetric external Diffie-Hellman (SXDH or 1-linear) assumption, which generally leads to most compact schemes. Instead, we use a variant of the decisional linear assumption (DLIN or 2-linear) on asymmetric groups (similar to [45], for example), which is generically no stronger than the same assumption on symmetric groups [17]—see §2.4 for details. Nonetheless, our CP- and KP-ABE schemes perform better than even the SXDH variant of CGW's schemes on almost all parameters of interest by operating primarily in the smaller and faster group $\mathbb{G}$.

2 PRELIMINARIES

We first define some notation that will be used throughout the paper. For a prime p, let $\mathbb{Z}_p$ denote the set $\{0, 1, 2, \ldots, p-1\}$ where addition and multiplication are done modulo p. The set $\mathbb{Z}_p^*$ is same as $\mathbb{Z}_p$ but with 0 removed.

Let λ denote the security parameter. $\mathsf{negl}(\lambda)$ denotes a negligible function, i.e., a function which is smaller than the inverse of any polynomial, for all large enough values of λ. A randomized algorithm is called PPT (probabilistic polynomial time) if its running time is bounded by some polynomial in the length of its input.

We use bold letters to denote vectors and matrices, with the former in lowercase and the latter in uppercase. By default, a vector must be treated as a column vector. $(\mathbf{v})_k$ denotes the kth element of a vector $\mathbf{v}$. $(\mathbf{M})_i$ and $(\mathbf{M})_{i,j}$ denote the ith row and the (i,j)th element of a matrix $\mathbf{M}$, respectively. We use $\mathbf{M}^{\top}$ for the transpose of

[4]In particular, the straightforward approach of generating an integer and mapping to a group element (via a generator) does not work. Instead, one should directly map the attributes to group elements.

$\mathbf{M}$. Also, $\langle \mathbf{a}, \mathbf{b} \rangle$ denotes the inner-product of vectors $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$, i.e., $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{n} a_i b_i$.

For any finite set S, we use $x \leftarrow_R S$ to denote that x is chosen uniformly at random from elements in S. Further, S^n denotes the set $\{(a_1, \ldots, a_n)^\top \mid a_i \in S \text{ for } i = 1, \ldots, n\}$ and, similarly, $S^{n \times m}$ denotes the set of matrices with n rows and m columns, each of whose elements lie in S.

Finally, $y \leftarrow \mathsf{Alg}(x)$ denotes that y is the output of running algorithm Alg on input x with uniformly random bits.

2.1 Access structures

An access structure or policy specifies the set of attributes required to gain access to some secret. More formally,

Definition 2.1 (Access structure). If $\mathcal{U}$ denotes the universe of attributes, then an access structure $\mathbb{A}$ is a collection of non-empty subsets of $\mathcal{U}$, i.e., $\mathbb{A} \subseteq 2^{\mathcal{U}} \setminus \{0\}$. It is called monotone if for every $B, C \subseteq \mathcal{U}$ such that $B \subseteq C$, $B \in \mathbb{A} \Rightarrow C \in \mathbb{A}$.

Monotonicity captures the natural idea that if an authorized user acquires more attributes, he/she cannot lose her privileges because of that.

A natural way to think about access control is in terms of (monotone) **Boolean formulae** with AND and OR gates, where each input is associated with an attribute in $\mathcal{U}$. A set of attributes $S \subseteq \mathcal{U}$ *satisfies* a formula if it evaluates to true on setting all inputs that map to some attribute in S to true, and the rest to false.

Boolean formulae fall into a more general class of functions called **monotone span programs** (MSPs) (or linear secret sharing schemes [50]). An MSP is given by a matrix $\mathbf{M}$ of size $n_1 \times n_2$ over $\mathbb{Z}_p$ and a mapping $\pi : \{1, \ldots, n_1\} \to \mathcal{U}$. In [44], Lewko and Waters describe a simple and efficient method to convert any (monotone) Boolean formula F into an MSP $(\mathbf{M}, \pi)$ such that every row of $\mathbf{M}$ corresponds to an input in F and the number of columns is same as the number of AND gates in F. Furthermore, each entry in $\mathbf{M}$ is either a 0, 1 or -1.[5]

Let S be a set of attributes and $I = \{i \mid i \in \{1, \ldots, n_1\}, \pi(i) \in S\}$ be the set of rows in $\mathbf{M}$ that *belong* to S. We say that $(\mathbf{M}, \pi)$ *accepts* S if there exists a linear combination of rows in I that gives $(1, 0, \ldots, 0)$. More formally, there should exist coefficients $\{\gamma_i\}_{i \in I}$ such that

$$\sum_{i \in I} \gamma_i (\mathbf{M})_i = (1, 0, \ldots, 0), \tag{2.1}$$

where $(\mathbf{M})_i$ is the ith row of $\mathbf{M}$. It is worth nothing that if Lewko and Water's method is applied on Boolean formulas, then it is always possible to pick coefficients that are either 0 or 1 for the resulting MSPs, irrespective of the set S.

Finally we state a lemma that will be useful in the security analysis of our ABE schemes. (See [13, Claim 2] for a proof.)

Lemma 2.2. *If an MSP* $(\mathbf{M}, \pi)$ *is not satisfied by a set of attributes* S*, then there exists a vector* $\mathbf{w}$ *whose first entry is non-zero and* $\forall\, i$ *such that* $\pi(i) \in S$, $\langle \mathbf{w}, (\mathbf{M})_i \rangle = 0$.

[5] If a formula has general k-out-of-n threshold gates, then $\mathbf{M}$'s entries may have a larger range. (A threshold gate evaluates to true if any of the k out of n inputs are true. Hence, OR is a 1-out-of-2 gate and AND is a 2-out-of-2 gate.)

2.2 Ciphertext-policy ABE

A ciphertext-policy ABE scheme over a message space $\mathcal{M}$ is given by four algorithms that behave as follows:

- $\mathsf{Setup}(1^\lambda)$. Given the security parameter λ as input, it outputs a public key pk and a master secret key msk.
- $\mathsf{Encrypt}(\mathsf{pk}, \mathbb{A}, \mathsf{msg})$. On input the public key pk, an access structure $\mathbb{A}$ (in the form of a Boolean formula, MSP, etc.), and a message $\mathsf{msg} \in \mathcal{M}$, it outputs a ciphertext ct.
- $\mathsf{KeyGen}(\mathsf{msk}, S)$. On input the master secret key msk and a set of attributes S, it outputs a secret key sk.
- $\mathsf{Decrypt}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk})$. On input the public key pk, a ciphertext ct, and a secret key sk, it outputs a message $\mathsf{msg}^* \in \mathcal{M}$ or a special symbol $\perp$.

Even though not explicitly stated, every algorithm above receives λ as input, and must run in $\mathsf{poly}(\lambda)$ time. They must also satisfy the following **correctness** condition: For all messages $\mathsf{msg} \in \mathcal{M}$, access structures $\mathbb{A}$, and set of attributes S that lie in $\mathbb{A}$, and for all $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda)$, $\Pr[\mathsf{Decrypt}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk}) \neq \mathsf{msg}] \leq \mathsf{negl}(\lambda)$, where $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pk}, \mathbb{A}, \mathsf{msg})$ and $\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, S)$. (Decrypt is assumed to be deterministic w.l.o.g.)

We assume that ciphertexts and keys also contain a description of the access structure and set of attributes, respectively, that they encode. But since in practice the description size will be much smaller compared to the cryptographic part, we do not consider it any further.

2.3 IND-CPA security

Intuitively, an ABE scheme is secure against chosen plaintext attacks (CPA) if no group of colluding users can distinguish between encryption of m_0 and m_1 under an access structure $\mathbb{A}^\star$ of their choice as long as no member of the group is authorized to decrypt on his/her own. Such attacks could occur any time after the deployment of ABE scheme. Thus the choice of $\mathbb{A}^\star$ is influenced by the public parameters and the keys in possession of the colluding users. When this is taken into account, one gets *adaptive* or *full* security. On the other hand, a weaker notion called *selective* security only prevents CPA attacks when $\mathbb{A}^\star$ is chosen even before the system is deployed, which is unlikely to happen in practice.

Adaptive security for an ABE scheme Π is formally defined with the help of a game $\mathsf{Expt}_{\Pi, \mathcal{A}}(\lambda, b)$ between a challenger Chal and an adversary $\mathcal{A}$, where Chal gets both 1^λ and b, and $\mathcal{A}$ gets 1^λ.

- (setup.) Chal runs $\mathsf{Setup}(1^\lambda)$ of Π to obtain pk and msk, and gives pk to $\mathcal{A}$.
- (key query.) $\mathcal{A}$ sends a set of attributes S. Chal then runs $\mathsf{KeyGen}(\mathsf{msk}, S)$ to obtain a key, which is returned to $\mathcal{A}$. This step is repeated as many times as $\mathcal{A}$ desires.
- (challenge.) $\mathcal{A}$ submits two messages $\mathsf{msg}_0, \mathsf{msg}_1$ and an access structure $\mathbb{A}^\star$. Chal then runs $\mathsf{Encrypt}(\mathsf{pk}, \mathbb{A}^\star, \mathsf{msg}_b)$ to get a ciphertext, which is returned to $\mathcal{A}$.
- (key query.) This phase is same as the second one.

$\mathcal{A}$ outputs a bit at the end of the game, which is defined to be the game's output. It is required that for every S queried by $\mathcal{A}$, $S \notin \mathbb{A}^\star$ (otherwise, b can be trivially guessed).

Definition 2.3. A CP-ABE scheme Π is called *fully* or *adaptively* secure if for all PPT adversaries $\mathcal{A}$,

$$\mathsf{Adv}_{\Pi}^{\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Expt}_{\Pi,\mathcal{A}}(\lambda,0)=1] - \Pr[\mathsf{Expt}_{\Pi,\mathcal{A}}(\lambda,1)=1]\right|$$

is negligible in λ.

2.4 Bi-linear maps and assumption

A map or pairing f from two source groups $\mathbb{X}$ and $\mathbb{Y}$ to a target group $\mathbb{T}$, all three of them multiplicative and of size ℓ, is called bi-linear if for all $a, b \in \mathbb{Z}$, $x \in \mathbb{X}$, $y \in \mathbb{Y}$, it holds that $f(x^a, y^b) = f(x,y)^{ab}$. Further, f is non-degenerate if $f(x,y) = 1$ implies that either $x = 1$ or $y = 1$. A pairing is asymmetric or Type-III if no efficiently computable homomorphism exists between the two source groups [25].

Let GroupGen be an asymmetric pairing group generator that on input 1^λ, outputs description of three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order $p = \Theta(\lambda)$ equipped with a non-degenerate efficiently computable bi-linear map $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. It also outputs generators g and h for $\mathbb{G}$ and $\mathbb{H}$, respectively.

Definition 2.4 (Decisional linear assumption). An asymmetric pairing group generator GroupGen satisfies the decisional linear assumption (DLIN) if for all PPT adversaries $\mathcal{A}$,

$$\mathsf{Adv}_{\mathsf{DLIN}}^{\mathcal{A}}(\lambda) := \Big|\Pr[\mathcal{A}(1^\lambda, \mathsf{par}, D, T_0) = 1] - \Pr[\mathcal{A}(1^\lambda, \mathsf{par}, D, T_1) = 1]\Big|$$

is negligible in λ, where $\mathsf{par} := (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h) \leftarrow \mathsf{GroupGen}(1^\lambda)$; $a_1, a_2 \leftarrow_R \mathbb{Z}_p^*$; $s_1, s_2, s \leftarrow_R \mathbb{Z}$; $D := (g^{a_1}, g^{a_2}, h^{a_1}, h^{a_2}, g^{a_1 s_1}, g^{a_2 s_2}, h^{a_1 s_1}, h^{a_2 s_2})$; $T_0 := (g^{s_1+s_2}, h^{s_1+s_2})$; $T_1 := (g^s, h^s)$.

We point out that there are different versions of DLIN on asymmetric groups. In sDLIN, an adversary is given *half* of the terms from above: either all from $\mathbb{G}$ or from $\mathbb{H}$ [26]. Sometimes a mix of terms from the two groups is used [30]. Our version is most similar to the one in Libert et al. [45, Appendix I]. We need such an assumption because our ciphertexts and keys have elements from both the groups, which is why SXDH does not work. Note that it is generically no stronger than the DLIN assumption on symmetric groups [17].

One can also define a variant of the k-linear family of assumptions [54] in a manner similar to Definition 2.4. Our ABE schemes can in fact be generalized to work for any $k \geq 2$ and the security would then follow from the corresponding assumption. However, the schemes' complexity would roughly increase linearly with k. We have implemented the general version of the scheme [2] but consider only the most efficient one here (for $k = 2$).

2.5 Representing group elements

Following [19, 22], we use $[x]_1$, $[y]_2$ and $[z]_T$ to denote g^x, h^y and $e(g,h)^z$, respectively, for $g \in \mathbb{G}$ and $h \in \mathbb{H}$ which will be clear from context. If $\mathbf{v}$ is a vector given by $(v_1, v_2, \ldots, v_n)^\mathsf{T}$ then $[\mathbf{v}]_1$ means $(g^{v_1}, g^{v_2}, \ldots, g^{v_n})^\mathsf{T}$. $[\mathbf{M}]_1$ for a matrix $\mathbf{M}$ is defined similarly. These operations are defined in the groups $\mathbb{H}$ and $\mathbb{G}_T$ in an analogous manner. Finally, $e([\mathbf{A}]_1, [\mathbf{B}]_2)$ for two matrices $\mathbf{A}$, $\mathbf{B}$ is defined as $[\mathbf{A}^\mathsf{T}\mathbf{B}]_T$.

Observe that given $[\mathbf{U}]_1$ for a matrix $\mathbf{U}$ of size $a \times b$, it is straightforward to compute $[\mathbf{UV}]_1$ for any matrix $\mathbf{V}$ of size $b \times c$ by doing abc exponentiations and $ac(b-1)$ multiplications in the worst case.

If we define $\mathbf{A}$, $\mathbf{s}$ and $\mathbf{s}'$ to be

$$\begin{bmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} s_1 \\ s_2 \\ s \end{bmatrix},$$

respectively, then using the notation just described, we can state the DLIN assumption succinctly as

$$([\mathbf{A}]_1, [\mathbf{A}]_2, [\mathbf{As}]_1, [\mathbf{As}]_2) \approx ([\mathbf{A}]_1, [\mathbf{A}]_2, [\mathbf{s}']_1, [\mathbf{s}']_2),$$

where $\approx$ denotes computational indistinguishability. (It is implicit that an adversary also gets par as an input.) We use this succinct version in the rest of the paper.

3 FAME: OUR CP-ABE SCHEME

In this section, we give a formal description of our ciphertext-policy ABE scheme Fame. The scheme uses a hash function $\mathcal{H}$ which maps arbitrary binary strings to elements of the group $\mathbb{G}$. In the security proof, $\mathcal{H}$ will be modeled as a random oracle.

Please note that the description of Fame is not intended to make the connections to CGW [19] explicit. In fact, we refrain from using the shorthand for group representation (widely used in CGW and described in Section 2.5) at this point so that the reader can quickly estimate the complexity of the scheme in terms of the size of each component, number of operations required to compute them, etc. When we set out to prove security of Fame afterwards (Section 4), we will present an alternate formulation of its algorithms along the lines of CGW by re-interpreting the outputs of random oracle.

In Fame, two types of inputs will be given to $\mathcal{H}$: inputs of the form (x, ℓ, t) or that of the form (j, ℓ, t), where x is an arbitrary string, j is a positive integer, $\ell \in \{1,2,3\}$ and $t \in \{1,2\}$. For simplicity, we represent these two inputs as $x\ell t$ and $0j\ell t$, respectively, appending 0 at the beginning of the second one so that it is not confused with the first. We assume that the inputs are appropriately encoded so that no two different tuples *collide*. Figure 3.1 describes the scheme.

There are several points to note about Fame. First, every ciphertext and key has elements from both $\mathbb{G}$ and $\mathbb{H}$. (As far as we know, this feature is unique to our scheme.) In particular, ct_0 has 3 elements from $\mathbb{H}$, $\mathsf{ct}_1, \ldots, \mathsf{ct}_{n_1}$ have 3 elements each from $\mathbb{G}$, and ct' has one element from $\mathbb{G}_T$. (Though the time taken to generate a ciphertext depends on the number of columns n_2 in $\mathbf{M}$, the size of the ciphertext does not.) Also, sk_0 has 3 elements from $\mathbb{H}$ and sk_y, sk' have 3 elements each from $\mathbb{G}$, for all $y \in S$. Thus, our scheme is mainly comprised of elements from $\mathbb{G}$ and the time taken to generate ciphertexts and keys is determined by the cost of group operations in $\mathbb{G}$.

Also observe that the decryption procedure is doing only 6 pairing operations, but a large number of exponentiations in the source groups. Fortunately, all these exponentiations are in the faster group $\mathbb{G}$, thus bringing down the decryption time considerably. Moreover, if we use Lewko-Waters' approach to convert Boolean formulae into MSPs (as discussed in §2.1) then the reconstruction coefficients γ_i are either 0 or 1. As a result, there will be *no* exponentiations at all during decryption—just multiplications in $\mathbb{G}$.

- Setup(1^λ) Run GroupGen(1^λ) to obtain $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h)$. Pick $a_1, a_2 \leftarrow_R \mathbb{Z}_p^*$ and $d_1, d_2, d_3 \leftarrow_R \mathbb{Z}_p$. Output
$$(h, H_1 := h^{a_1}, H_2 := h^{a_2}, T_1 := e(g,h)^{d_1 a_1 + d_3}, T_2 := e(g,h)^{d_2 a_2 + d_3})$$
as the public key pk. Also, pick $b_1, b_2 \leftarrow_R \mathbb{Z}_p^*$ and output
$$(g, h, a_1, a_2, b_1, b_2, g^{d_1}, g^{d_2}, g^{d_3})$$
as the master secret key msk.
- KeyGen(msk, S) Pick $r_1, r_2 \leftarrow_R \mathbb{Z}_p$ and compute
$$\mathsf{sk}_0 \quad := \quad (h^{b_1 r_1}, h^{b_2 r_2}, h^{r_1 + r_2})$$
using h, b_1, b_2 from msk. For all $y \in S$ and $t = 1, 2$, compute
$$\mathsf{sk}_{y,t} \quad := \quad \mathcal{H}(y1t)^{\frac{b_1 r_1}{a_t}} \cdot \mathcal{H}(y2t)^{\frac{b_2 r_2}{a_t}} \cdot \mathcal{H}(y3t)^{\frac{r_1 + r_2}{a_t}} \cdot g^{\frac{\sigma_y}{a_t}},$$
where $\sigma_y \leftarrow_R \mathbb{Z}_p$. Set $\mathsf{sk}_y := (\mathsf{sk}_{y,1}, \mathsf{sk}_{y,2}, g^{-\sigma_y})$. Also, compute
$$\mathsf{sk}'_t \quad := \quad g^{d_t} \cdot \mathcal{H}(011t)^{\frac{b_1 r_1}{a_t}} \cdot \mathcal{H}(012t)^{\frac{b_2 r_2}{a_t}} \cdot \mathcal{H}(013t)^{\frac{r_1 + r_2}{a_t}} \cdot g^{\frac{\sigma'}{a_t}}$$
for $t = 1, 2$, where $\sigma' \leftarrow_R \mathbb{Z}_p$. Set $\mathsf{sk}' = (\mathsf{sk}'_1, \mathsf{sk}'_2, g^{d_3} \cdot g^{-\sigma'})$. Output $(\mathsf{sk}_0, \{\mathsf{sk}_y\}_{y \in S}, \mathsf{sk}')$ as the key.
- Encrypt(pk, $(\mathbf{M}, \pi)$, msg) Pick $s_1, s_2 \leftarrow_R \mathbb{Z}_p$. Compute
$$\mathsf{ct}_0 \quad := \quad (H_1^{s_1}, H_2^{s_2}, h^{s_1 + s_2})$$
using pk. Suppose $\mathbf{M}$ has n_1 rows and n_2 columns. Then, for $i = 1, \ldots, n_1$ and $\ell = 1, 2, 3$, compute
$$\mathsf{ct}_{i,\ell} \quad := \quad \mathcal{H}(\pi(i)\ell 1)^{s_1} \cdot \mathcal{H}(\pi(i)\ell 2)^{s_2} \cdot \prod_{j=1}^{n_2} \left[\mathcal{H}(0j\ell 1)^{s_1} \cdot \mathcal{H}(0j\ell 2)^{s_2}\right]^{(\mathbf{M})_{i,j}},$$
where, recall that, $(\mathbf{M})_{i,j}$ denotes the (i,j)th element of $\mathbf{M}$. Set $\mathsf{ct}_i := (\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}, \mathsf{ct}_{i,3})$. Also, compute
$$\mathsf{ct}' \quad := \quad T_1^{s_1} \cdot T_2^{s_2} \cdot \mathsf{msg}.$$
Output $(\mathsf{ct}_0, \mathsf{ct}_1, \ldots, \mathsf{ct}_{n_1}, \mathsf{ct}')$ as the ciphertext.
- Decrypt(pk, ct, sk) Recall that if the set of attributes S in sk satisfies the MSP $(\mathbf{M}, \pi)$ in ct, then there exists constants $\{\gamma_i\}_{i \in I}$ that satisfy (2.1). Now, compute
$$\mathsf{num} \quad := \quad \mathsf{ct}' \cdot e\left(\prod_{i \in I} \mathsf{ct}_{i,1}^{\gamma_i}, \mathsf{sk}_{0,1}\right) \cdot e\left(\prod_{i \in I} \mathsf{ct}_{i,2}^{\gamma_i}, \mathsf{sk}_{0,2}\right) \cdot e\left(\prod_{i \in I} \mathsf{ct}_{i,3}^{\gamma_i}, \mathsf{sk}_{0,3}\right),$$
$$\mathsf{den} \quad := \quad e\left(\mathsf{sk}'_1 \cdot \prod_{i \in I} \mathsf{sk}_{\pi(i),1}^{\gamma_i}, \mathsf{ct}_{0,1}\right) \cdot e\left(\mathsf{sk}'_2 \cdot \prod_{i \in I} \mathsf{sk}_{\pi(i),2}^{\gamma_i}, \mathsf{ct}_{0,2}\right) \cdot e\left(\mathsf{sk}'_3 \cdot \prod_{i \in I} \mathsf{sk}_{\pi(i),3}^{\gamma_i}, \mathsf{ct}_{0,3}\right),$$
and output num/den. Here $\mathsf{sk}_{0,1}, \mathsf{sk}_{0,2}, \mathsf{sk}_{0,3}$ denote the first, second and third elements of sk_0; the same for ct_0.

Figure 3.1: FAME: ciphertext-policy attribute-based encryption.

Please see Appendix A for the correctness of FAME. We now discuss some issues pertinent to the use of ABE schemes.

Encrypting large messages. As the reader may have noticed, the plaintext data given to the encryption algorithm in FAME is an element of the target group. In practice this data would be too large to be encoded as a single element of $\mathbb{G}_T$, and it would be very expensive to break it into small pieces and ABE encrypt each piece separately. The standard method is to use a key encapsulation mechanism (KEM) wherein a random element of $\mathbb{G}_T$ is ABE encrypted and hashed to derive a session key. This key is then used to encrypt the plaintext data through a fast symmetric key scheme like AES. Thus, the overhead of encrypting any amount of data via an ABE scheme is reduced to the cost of just one application of ABE encrypt. An even more efficient variant would simply hash $T_1^{s_1} T_2^{s_2}$ and use the result as the symmetric key—a very similar proof to the one for FAME would show that this is a secure ABE-KEM.

One-use restriction. As is true for all known fully secure schemes secure under standard assumptions, our scheme requires the mapping π in an MSP to be an injective function, i.e., no two rows should be mapped to the same attribute. This is commonly referred to as the one-use restriction. [6] A common way of getting around this problem, as suggested in many papers like [42, 57], is to have k copies of each attribute in the universe for some fixed k chosen at set-up. For example, 'Title:Prof' will be replaced by 'Title:Prof:1', 'Title:Prof:2', ..., 'Title:Prof:k'. The downside of this transformation is that the size of keys grows by a factor of k; but note that the encryption and decryption time is not affected.[7]

[6]Kowalczyk and Lewko KP-ABE schemes [40] also have one-use restriction. The public parameters in their scheme grow logarithmically rather than linearly in the bound on attribute re-use, but ciphertexts still grow linearly. Their prime-order construction was broken and has been removed from the full version.

[7]We could modify FAME to prevent a multiplicative increase in key-size by borrowing ideas from the unbounded attribute re-use scheme in [5], but the security assumption

Non-monotonicity. Though monotonicity is a very natural property for access structures (Section 2.1), non-monotonic policies can also be useful. For example, a CS department may want to make a certain set of files accessible to everybody except graduate students. FAME can be made to support such policies by introducing new attributes like 'Title:Not-Grad', but the problem is that a professor in the department, for instance, must now get all attributes of the type 'Title:Not-*', which could result in much larger keys. There are only a handful of schemes in literature that support non-monotonic access structures directly, with Ostrovsky et al. [48] KP-ABE being the most popular one. Though these schemes are able to avoid the 'Title:Not-*' problem, they also *fix* the number of attributes *any* ciphertext must have and require that the entire ciphertext be used in every decryption (so that a user cannot *pretend* not to have a certain attribute), thus resulting in larger ciphertexts and slower decryption.

4 SECURITY OF FAME

The security proof proceeds via a series of hybrids. A hybrid describes how the challenger Chal interacts with an adversary $\mathcal{A}$. The *zeroth* hybrid, Hyb_0, is of course the one where Chal and $\mathcal{A}$ interact according to $\mathsf{Expt}_{\Pi,\mathcal{A}}(1^\lambda, b)$ (§2.3) with Π being our scheme FAME. The only difference is that the hash function $\mathcal{H}$ is assumed to behave like a random oracle.

The first step in the security analysis is to rewrite FAME in a compact form by interpreting the outputs of random oracle appropriately and using the notation defined in §2.4 to represent group elements. This compact form will be the first hybrid, Hyb_1. Here one can see the connections to CGW more clearly.

The compact form also simplifies rest of the proof presentation. So we discuss Hyb_1 at length first and give a high-level overview of the proof after that.

4.1 Compact representation

Let Samp be an algorithm that on input a prime p, outputs

$$\mathbf{Z} := \begin{bmatrix} u_1 & 0 \\ 0 & u_2 \\ 1 & 1 \end{bmatrix}, \qquad \mathbf{z}^\perp := \begin{bmatrix} u_1{}^{-1} \\ u_2{}^{-1} \\ -1 \end{bmatrix}, \tag{4.1}$$

where $u_1, u_2 \leftarrow_R \mathbb{Z}_p^*$. Appendix C.2 discusses some interesting properties of this algorithm.

We define a modified version of the IND-CPA game $\mathsf{Expt}_{\text{FAME},\mathcal{A}}(1^\lambda, b)$, called Hyb_1, in this section. To begin with, the challenger Chal sets up the ABE scheme as follows:

Setup. Run GroupGen(1^λ) to obtain $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h)$ as before. Pick $(\mathbf{A}, \mathbf{a}^\perp)$, $(\mathbf{B}, \mathbf{b}^\perp) \leftarrow \mathsf{Samp}(p)$ and $d_1, d_2, d_3 \leftarrow_R \mathbb{Z}_p$. Let $\mathbf{d}$ denote the column vector $(d_1, d_2, d_3)^\top$. Set $\mathsf{pk} := ([\mathbf{A}]_2, [\mathbf{d}^\top\mathbf{A}]_T)$, $\mathsf{msk} := (g, h, \mathbf{A}, \mathbf{B}, [\mathbf{d}]_1)$.

In order to simulate the random oracle, Chal maintains two lists L and Q. The list L has entries of the form $(x, \mathbf{W}_x)$ or $(j, \mathbf{U}_j)$ where x is an arbitrary binary string, j is a positive integer, and $\mathbf{W}_x, \mathbf{U}_j$ are 3×3 matrices over $\mathbb{Z}_p$.[8] The list Q has entries of the form (q, r) where q is either $x\ell t$ or $0j\ell t$ (for $\ell \in \{1,2,3\}$ and $t \in \{1,2\}$) or something else, and r is an element of $\mathbb{G}$.

Adversary $\mathcal{A}$ can make one of three types of oracle queries:

(1) $x\ell t$: Chal checks if $(x\ell t, r) \in Q$ for some r or not. If such an entry is found then it returns r, otherwise it checks if $(x, \mathbf{W}_x) \in L$ for some $\mathbf{W}_x$ or not. If such an entry is found then $r := [(\mathbf{W}_x^\top\mathbf{A})_{\ell,t}]_1$ is computed, $(x\ell t, r)$ is added to Q, and r is returned. Else, it picks $\mathbf{W}_x \leftarrow_R \mathbb{Z}_p^{3\times3}$, adds $(x, \mathbf{W}_x)$ to L, computes $r := [(\mathbf{W}_x^\top\mathbf{A})_{\ell,t}]_1$, adds $(x\ell t, r)$ to Q, and returns r.

(2) $0j\ell t$: Chal checks if $(0j\ell t, r) \in Q$ for some r or not. If such an entry is found then it returns r, otherwise it checks if $(j, \mathbf{U}_j) \in L$ for some $\mathbf{U}_j$ or not. If such an entry is found then $r := [(\mathbf{U}_j^\top\mathbf{A})_{\ell,t}]_1$ is computed, $(0j\ell t, r)$ is added to Q, and r is returned. Else, it picks $\mathbf{U}_j \leftarrow_R \mathbb{Z}_p^{3\times3}$, adds $(x, \mathbf{U}_j)$ to L, computes $r := [(\mathbf{U}_j^\top\mathbf{A})_{\ell,t}]_1$, adds $(0j\ell t, r)$ to Q, and returns r.

(3) Anything else, say q: Chal checks if $(q, r) \in Q$ for some r or not. If such an entry is found then it returns r, otherwise a random element from $\mathbb{G}$, say r', is picked, (q, r') is added to Q, and r' is returned.

Key generation. When $\mathcal{A}$ makes a key query S, Chal retrieves $\mathbf{W}_y$ for every $y \in S$ and $\mathbf{U}_1$ from the list L. (If one of them is not available then a random 3×3 matrix is generated like above. The list L is also updated accordingly.) Now pick $r_1, r_2, \sigma' \leftarrow_R \mathbb{Z}_p$ as well as $\sigma_y \leftarrow_R \mathbb{Z}_p$ for $y \in S$. Let $\mathbf{r} = (r_1, r_2)^\top$ and compute

$$\mathsf{sk}_0 := [\mathbf{Br}]_2, \qquad \mathsf{sk}_y := [\mathbf{W}_y\mathbf{Br} + \sigma_y\mathbf{a}^\perp]_1,$$
$$\mathsf{sk}' := [\mathbf{d} + \mathbf{U}_1\mathbf{Br} + \sigma'\mathbf{a}^\perp]_1$$

for all $y \in S$. Then return $(\mathsf{sk}_0, \{\mathsf{sk}_y\}_{y\in S}, \mathsf{sk}')$ as the key.

Encryption. When $\mathcal{A}$ sends messages $\mathsf{msg}_0, \mathsf{msg}_1$ and a policy $(\mathbf{M}, \pi)$, Chal retrieves $[(\mathbf{W}_{\pi(i)}^\top\mathbf{A})_{\ell,t}]_1$ and $[(\mathbf{U}_j^\top\mathbf{A})_{\ell,t}]_1$ for all $i = 1, \ldots, n_1$, $j = 1, \ldots, n_2$, ℓ, t from the list Q. (If a $\pi(i)\ell t$ or $0j\ell t$ is not found in Q, then it follows the same process as in (1) or (2) above, respectively.) Now pick $s_1, s_2 \leftarrow_R \mathbb{Z}_p$, set $\mathbf{s}$ to be (s_1, s_2), and compute

$$\mathsf{ct}_0 := [\mathbf{As}]_2, \qquad \mathsf{ct}_i := \left[\mathbf{W}_{\pi(i)}^\top\mathbf{As} + \sum_{j=1}^{n_2}(\mathbf{M})_{i,j}\mathbf{U}_j^\top\mathbf{As}\right]_1$$
$$\mathsf{ct}' := [\mathbf{d}^\top\mathbf{As}]_T \cdot \mathsf{msg}_b,$$

for $i = 1, \ldots, n_1$. Return ciphertext $(\mathsf{ct}_0, \mathsf{ct}_1, \ldots, \mathsf{ct}_{n_1}, \mathsf{ct}')$.

4.2 High-level overview

Even though Hyb_0, with the algorithms of FAME, looks very different from Hyb_1, they are in fact identical from the point of view of any adversary. At a high level, the $\mathbf{W}_x, \mathbf{U}_j$ matrices have enough entropy to make $(\mathbf{W}_x^\top\mathbf{A})_{\ell,t}$, $(\mathbf{U}_j^\top\mathbf{A})_{\ell,t}$ look random for every ℓ, t. Further, when the hashed values in the ciphertexts/keys of FAME are interpreted in the way the challenger simulates them, one can then carefully manipulate them to show that they match with those in Hyb_1.

The structure of ciphertexts and keys in Hyb_1 appears similar to that of CGW's CP-ABE scheme [19, Appendix B.2]. One clear

would have to be parameterized by the degree d of attribute reuse, and the number of pairings required for decryption would also increase by a factor of d.

[8] Assume that the x and j are appropriately encoded so that they don't collide.

and important difference is that while our ciphertexts and keys have only the first component in group $\mathbb{H}$, theirs are composed entirely of elements from $\mathbb{G}$ and $\mathbb{H}$, respectively. From a security perspective, we have an additional $\mathbf{a}^\perp$ component in our keys that is not present in theirs. We define a sequence of hybrids, called Group-I hybrids, to get rid of this component. These hybrids are specific to our proof.

Group-I has $3Q$ hybrids from $\mathsf{Hyb}_{2,1,1}$ to $\mathsf{Hyb}_{2,3,Q}$, where Q is the number of key queries an adversary makes. These hybrids modify the key components one by one. First, DLIN is used to replace $\mathbf{Br}$ by $\mathbf{Br} + \hat{r}\mathbf{a}^\perp$ (Definition 2.4, §2.5) for a random $\hat{r}$ because the linear independence of $\mathbf{a}^\perp$ from $\mathbf{B}$ (Lemma C.1) makes $\mathbf{Br} + \hat{r}\mathbf{a}^\perp$ a random vector. Second, the $\mathbf{W}_x$ matrices have one unit of residual entropy even given $\mathbf{W}_x^\top\mathbf{A}$ and $\mathbf{W}_x\mathbf{B}$ (same with $\mathbf{U}_j$), which can be exploited to *absorb* the extra $\mathbf{a}^\perp$ component without affecting the challenge ciphertext and other parts of the keys. This type of information-theoretic step is usually called parameter-hiding in dual-system encryption based proofs [19, 56]. Lastly, DLIN is used to revert back to $\mathbf{Br}$.

We then define another set of hybrids, called Group-II hybrids, to show that the encryption of any message is indistinguishable from the encryption of a random message. Group-II has $3Q + 2$ hybrids: Hyb_3, $\mathsf{Hyb}_{4,1,1}, \ldots, \mathsf{Hyb}_{4,3,Q}$, and Hyb_5. The first among them, Hyb_3, uses DLIN to replace $\mathbf{As}$ by $\mathbf{As} + \hat{s}\mathbf{b}^\perp$ in the challenge ciphertext, possible again due to linear independence. The new form of ciphertext is called semi-functional, a term first used by Waters [56]. The sequence from $\mathsf{Hyb}_{4,1,1}$ to $\mathsf{Hyb}_{4,3,Q}$ is somewhat similar to $\mathsf{Hyb}_{2,1,1}$ to $\mathsf{Hyb}_{2,3,Q}$ in terms of the changes made to key components. The residual entropy in $\mathbf{W}_x, \mathbf{U}_j$ is used towards a different purpose now: to *introduce* some structured randomness into the key.

Moving from $\mathsf{Hyb}_{4,1,1}$ to $\mathsf{Hyb}_{4,3,Q}$ requires more care because the ciphertext is semi-functional. We must make sure that while the keys are being transformed, the ciphertext can still be generated given just a DLIN tuple. Furthermore, the parameter-hiding step affects not only the keys but the ciphertext too. At this stage, we use the fact that none of the keys issued to the adversary can decrypt the challenge ciphertext.

$\mathsf{Hyb}_{4,3,Q}$ is almost the same as $\mathsf{Hyb}_{2,3,Q}$, the last of the Group-I hybrids, except that the ciphertext is semi-functional and the keys have some *extra* randomness. The last step, which leads to Hyb_5, *moves* this randomness to the ciphertext, so that it is indistinguishable from the encryption of a random message.

4.3 Main theorem

We now formally state the security property of Fame.

Theorem 4.1. *Fame, defined in Figure 3.1, is fully secure (Def 2.3) under the* DLIN *assumption on asymmetric pairing groups (Def 2.4) in the random oracle model. Concretely, for any* PPT *adversary $\mathcal{A}$ making Q key queries in the IND-CPA security game, there exists a* PPT *adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathcal{A}}_{FAME}(\lambda) \leq (8Q+2)\mathsf{Adv}^{\mathcal{B}}_{\mathsf{DLIN}}(\lambda) + (16Q+6)/p,$$

where $p = \Theta(\lambda)$ is the order of the pairing group.

A proof of the above theorem can be found in Appendix C. There, we first formally describe the hybrids that will be used in the proof, and how we go from one hybrid to the next (C.1). Then we show why a hybrid in the sequence is indistinguishable from the next one (C.3). And finally we prove the theorem with the help of these indistinguishable hybrids (C.4).

5 IMPLEMENTATION & EVALUATION

We implement ABE schemes in Python 2.7.10 using the Charm 0.43 framework [7]. We use MNT224 curve for pairings because it is the best Type-III curve in PBC, the default pairing library in Charm. It provides 96-bit security level [59]. All running times below were measured on a Macbook Pro laptop with a 2.7 GHz Intel Core i5 processor and 8GB RAM. The implementation code is available on GitHub [2].

Table 5.1 lists the average time taken by various operations on MNT224 in milliseconds. One can see that operations on group $\mathbb{H}$ are significantly more expensive than on $\mathbb{G}$, from 7 times for multiplication to as much as 775 times for hashing. Pairing is a very expensive operation too: if we put exponentiation and hashing in $\mathbb{H}$ aside then pairing is at least thrice as costly as any other operation. It is also important to note that the size of an element in $\mathbb{H}$ is 3 times that of $\mathbb{G}$. [9]

Groups	*Multiplication*	*Exponentiation*	*Hash*
$\mathbb{G}$	.009	1.266	.099
$\mathbb{H}$	.065	14.412	76.767
$\mathbb{G}_T$	.020	3.356	-

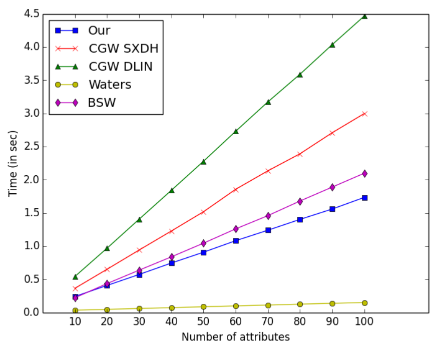
Pairing	10.243

Table 5.1: Average time taken by various operations on the MNT224 curve. Pairing operation is listed separately. All times are measured in *milliseconds* correct to three decimal places.

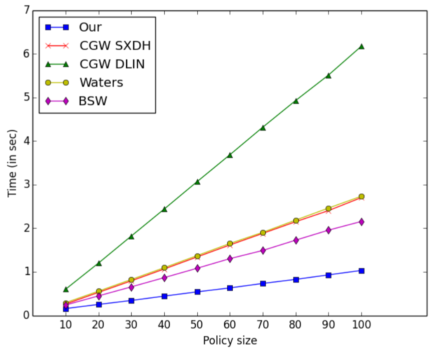
We use access policies of type 'Attr1 AND Attr2 AND ... AND AttrN' as in Green et al. [31] because all the N attributes are then required for decryption. We say that such a policy is of size N. We test all the schemes against policies and attribute sets of size $10, 20, \ldots, 100$. As argued by Green et al., large policy sizes are quite likely in typical use cases (e.g., a restriction window involving a Unix time value). We first convert the policies into a Boolean formula and then to an MSP using Lewko-Waters' method, the advantage being that the matrix generated has only 0, 1 or -1 entries and the reconstruction coefficients are always just 0 or 1 (see §2.1 for a detailed discussion.)

CP-ABE. Besides Fame, we implement Bethencourt et al.'s (BSW) [16, Section 4.2], Waters' [57, Section 3] and CGW's [20, Appendix B.2] CP-ABE schemes under the same setting. There are other implementations of the first two schemes but, as far we know, CGW's schemes have not been implemented before, nor have any other fully secure schemes. Below, we compare with both the SXDH (1-linear) and DLIN (2-linear) instantiations of CGW, the two assumptions under which it gives the best performance. They are referred to as CGW-1 and CGW-2, respectively, for brevity.

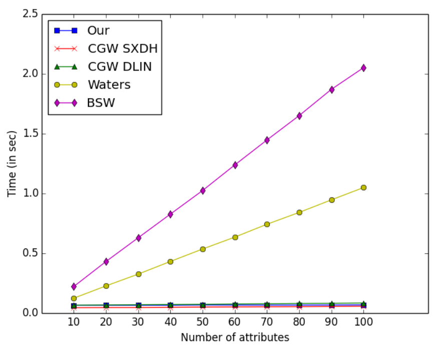
[9]Though the numbers here are specifically for the MNT224 curve, other Type-III curves like Bareto-Naehrig have similar disparity between groups $\mathbb{G}$ and $\mathbb{H}$ [33].



(a) Key generation

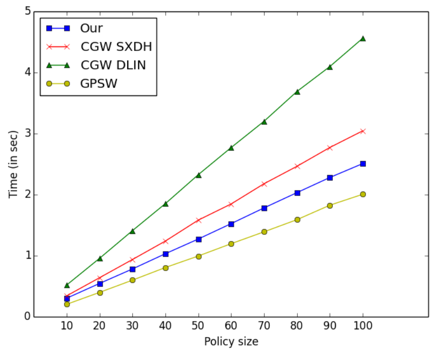


(b) Encryption

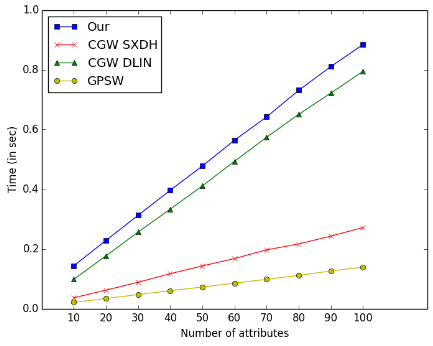


(c) Decryption

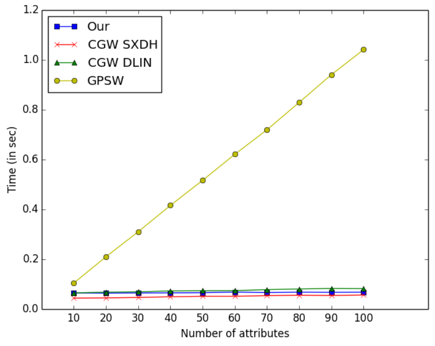
Figure 5.1: Ciphertext-policy attribute-based encryption.



(a) Key generation



(b) Encryption



(c) Decryption

Figure 5.2: Key-policy attribute-based encryption.

Scheme	*Uni size*	*Time*
Our	-	0.11s
CGW-1	100	2.23s
CGW-2	100	5.13s
Waters	100	0.64s
BSW	-	0.08s

Scheme	*Uni size*	*Time*
Our	-	0.11s
CGW-1	100	4.08s
CGW-2	100	9.33s
GPSW	100	0.64s

Figure 5.3: Set-up times for CP-ABE (left) and KP-ABE (right).

We chose BSW because it was the first scheme proposed for CP-ABE, and it is quite popular in the community for its simplicity and efficiency, but proved secure only under the generic group model (GGM). Waters' scheme was the first to be proved in the standard model, albeit only selectively. CGW is an obvious choice since it is the most efficient fully secure scheme (in the standard model) for a bounded universe of attributes.

It is worth noting that both BSW's and Waters' schemes were built using symmetric bilinear maps, which have serious security issues [24, 37]. We implement them in the asymmetric setting using the MNT224 curve (see Appendix D and E). In this process, the number of elements in ciphertexts or keys, and the number of group operations used in any of the algorithms is not affected. We believe that the modified version of Waters' scheme can be proved secure under a variant of the assumption he uses. Modifying BSW required more care because it uses a hash function. However, since BSW is proven secure in the generic group model, the security of the modified scheme is obvious (in the same model). [10]

Figure 5.3 (left) shows the time it takes to run the set-up algorithms of the CP-ABE schemes we implemented. For bounded universe schemes, we used the smallest bound possible, i.e., 100. The advantage of using an unbounded universe scheme is clear from this table: our scheme takes a mere one-tenth of a second to be set-up no matter how many attributes need to be supported. This is only slightly worse than BSW, another unbounded scheme only known to be secure in the generic group model, and substantially better than the best known fully secure scheme, even for a universe of size 100.

Figure 5.1 compares the running time of key generation, encryption and decryption algorithms for the CP-ABE schemes we

[10]Please note that there are various ways to convert a Type-I scheme to Type-III, and this process is already quite challenging for identity-based encryption [3, 6, 8]. We try to balance the total work fairly between encryption and key-generation (see Table 5.2), and avoid the use of expensive operations like hashing in $\mathbb{H}$ (see Table 5.1).

	Key generation						*Encryption*					
	$\mathbb{G}$			$\mathbb{H}$			$\mathbb{G}$			$\mathbb{H}$		
Schemes	Mul	Exp	Hash	Mul	Exp	Hash	Mul	Exp	Hash	Mul	Exp	Hash
Our	$8T+9$	$9T+9$	$6(T+1)$	-	3	-	$12n_1n_2+6n_1$	$6n_1$	$6(n_1+n_2)$	-	3	-
CGW-1	-	-	-	-	$2(T+2)$	-	$\sim 4n_1n_2$	$2n_1+4n_2$	-	-	-	-
CGW-2	-	-	-	-	$3(T+2)$	-	$\sim 6n_1n_2$	$6n_1+9n_2$	-	-	-	-
Waters	1	$T+1$	-	-	1	-	n_1	$2n_1$	-	-	n_1+1	-
BSW	$T+1$	$T+2$	T	-	T	-	-	n_1	n_1	-	n_1+1	-

Table 5.2: The number of various operations in $\mathbb{G}$ and $\mathbb{H}$ for key-generation and encryption in the implementations of CP-ABE schemes we consider. Here T denotes the number of attributes input to KeyGen**; and n_1, n_2 are the dimensions of the MSP input to** Encrypt**. The exact number for CGW-1 and CGW-2 multiplications in $\mathbb{G}$ are $2(n_1+2n_2+2n_1n_2-1)$ and $3(2n_1+3n_2+2n_1n_2-1)$, respectively.**

	Decryption			
	Multiplication			Pairing
Schemes	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{G}_T$	
Our	$6I+3$	-	6	6
CGW-1	$2I$	$2I$	4	4
CGW-2	$3I$	$3I$	6	6
Waters	I	-	3	$I+2$
BSW	-	-	$2I+1$	$2I+1$

Figure 5.4: The number of various operations in $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}_T$ for decryption in the implementations of CP-ABE schemes we consider. Here I is the number of attributes used in decryption.

	Key size		*Ciphertext size*	
Schemes	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{G}$	$\mathbb{H}$
Our	$3(T+1)$	3	$3n_1$	3
CGW-1	-	$2(T+2)$	$2(n_1+1)$	-
CGW-2	-	$3(T+2)$	$3(n_1+1)$	-
Waters	$T+1$	1	n_1	n_1+1
BSW	$T+1$	T	n_1	n_1+1

Figure 5.5: The size of ciphertexts and keys in the CP-ABE schemes we consider. '$\mathbb{G}$' and '$\mathbb{H}$' columns denote the number of elements in groups $\mathbb{G}$ and $\mathbb{H}$, respectively. T denotes the number of attributes input to KeyGen**; and n_1, n_2 are the dimensions of the MSP input to** Encrypt**. Note that the size of an element of $\mathbb{H}$ is 3 times that of $\mathbb{G}$ in the MNT224 curve.**

consider. Tables 5.2, 5.4 list the number of various group operations involved in the implementations of these algorithms. [11] [12]

Even though our scheme is based on the DLIN version of CGW, it outperforms even the SXDH version for key generation: when the number of attributes is 100, it takes roughly half the time of CGW SXDH. Only Waters' scheme does better but at the cost of much weaker security guarantees (selective security under a q-type assumption).

To understand why the schemes compare in this way, it is useful to study the key-generation column of Table 5.2. We can focus on the number of exponentiations because it is a lot more expensive than multiplication and hashing, see Table 5.1. (Hashing in $\mathbb{H}$ is most expensive but it is never used.) Our scheme has a total of about 4.5 times more exponentiations than CGW-1 and BSW but still performs better than both because we have found a way to do almost all the operations in the faster group $\mathbb{G}$. Waters' scheme does not have any operation in $\mathbb{H}$ (except one) and 9 times less exponentiations in $\mathbb{G}$, therefore it does better.

In terms of encryption time, we do better than *all* the other schemes: it takes just about a second to encrypt a policy of size 100! It is clear from Table 5.2 why Waters and BSW are worse: exponentiation in $\mathbb{H}$ is about 11 times slower than in $\mathbb{G}$. What is less clear is our better performance with respect to CGW, specifically CGW-1. This is because the randomness complexity of their encryption scheme is unusually high. As many as $4n_2$ random numbers need to be sampled for every encryption, and sampling needs much more time than hashing or multiplication (for the MNT224 curve in the Charm framework).

Perhaps the most striking aspect of our scheme is the decryption time. While it increases almost linearly for BSW and Waters' schemes with the number of attributes required to decrypt, both CGW's and our schemes always need just about 0.06 seconds! This is due to the fact that only a constant number of pairing operations are required. (The number of multiplication operations does grow linearly in all schemes according to Table 5.4 but that has no significant effect because even multiplication in $\mathbb{H}$ is about 150 times slower than pairing.)

Finally, we would like to draw the attention of the reader to Table 5.5 which lists the size of ciphertexts and keys in terms of the number of elements from $\mathbb{G}$ and $\mathbb{H}$. [13] A cursory look may give the impression that ciphertexts/keys of our scheme are not smaller than anyone else. However, recall that an element of $\mathbb{H}$ is 3 times as large as that of $\mathbb{G}$. So our key size is much smaller than all the schemes except Waters'; and ciphertext size is comparable to CGW-2 and smaller than both Waters and BSW.

KP-ABE. We briefly discuss the performance of our KP-ABE scheme (Appendix B). For comparison, we also implemented CGW's (SXDH

[11] An ABE ciphertext has a few target group elements that hide the message. The number of operations required to generate them have not been included in this table.

[12] If an entry of the MSP matrix is used in the exponent of an exponentiation operation, then we count the operation as a multiplication. Recall that the entries are either 0, 1 or −1 (§2.1), so even in the worst case there will be an inversion operation, which is faster than multiplication.

[13] An ABE ciphertext has a few target group elements that hide the message. They have not been included in this table.

and DLIN) [20, Appendix B.1] and Goyal et al.'s (GPSW) KP-ABE schemes [29, Appendix A.1]. Figure 5.3 (right) lists the set-up time and Figure 5.2 plots the time taken by other operations. Also see Appendix F for the asymmetric version of GPSW that we implemented.

Once again the set-up time is a very small constant, the decryption time is only about 0.06s, and key generation is better than CGW-1 (only about a second for a policy size of 40). Encryption time, though larger than other schemes, is no more than 0.9s for as many as 100 attributes.

Further improvements. There are a number of ways to further optimize the performance of our schemes. A natural idea is to use C/C++ instead of Python and interface directly with a pairing library (instead of using Charm's wrappers). The Charm framework, however, does have several benefits like pre-computation tables that significantly speed up exponentiations, which we have *not* exploited here. One could also take advantage of multi-exponentiation and products of pairings.

Another option would be to use a different curve for pairings, like the Bareto-Naehrig (BN) curves. Please note that there are attacks known on certain parameters for both MNT and BN curves [34, 39]. Hence one must choose a curve carefully for a real world application.

6 RELATED WORK

We discuss some related work in this section that has not been referred to or discussed in detail in the introduction.

A number of methods have been devised to translate schemes based on composite-order groups to the prime-order setting [23, 35, 41] but they are not general purpose. Moreover, the resulting schemes usually have a factor more group elements in the ciphertexts/keys than the original scheme.

Some sophisticated tools have been developed to automate the translation of Type-I to Type-III pairings [3, 6, 8] but they have been applied to (hierarchical) identity-based encryption, broadcast encryption and signature schemes only. It is not clear if the tools can handle more advanced encryption primitives like ABE.

Okamoto and Takashima have developed fully secure schemes under the DLIN assumption on symmetric maps which support a large number of attributes [47], but theirs is not a large-universe construction in the standard sense. They consider *(attribute, value)* pairs where each *attribute* takes a value from an exponential-sized space, instead of being *present* or *not present*. Their security proof requires a polynomial sized set of all possible attributes to be known in advance. Moreover, 14 group elements are needed in ciphertext/key for every attribute, and decryption is similarly slow. On the other hand, their approach makes it easy to handle non-monotonic policies where one could have conditions like an attribute should not have a particular value.

Attrapadung has recently proposed some large universe constructions on asymmetric maps [11] under q-type assumptions. Our use of random oracle not only eliminates such non-standard assumptions but also gives much more efficient constructions. For example, Attrapadung's unbounded KP-ABE scheme has ciphertexts with 6 group elements per attribute, keys with 9 elements per matrix row, and requires 9 pairings per attribute to decrypt, whereas our KP-ABE scheme (Figure B.1) does much better.

REFERENCES

[1] Zeutro LLC - Encryption and Data Security. http://www.zeutro.com/.
[2] Attribute-based Encryption. https://github.com/sagrawal87/ABE, 2017.
[3] M. Abe, J. Groth, M. Ohkubo, and T. Tango. Converting cryptographic schemes from symmetric to asymmetric bilinear groups. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 241–260. Springer, Heidelberg, Aug. 2014.
[4] S. Agrawal and M. Chase. A study of pair encodings: Predicate encryption in prime order groups. In E. Kushilevitz and T. Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 259–288. Springer, Heidelberg, Jan. 2016.
[5] S. Agrawal and M. Chase. Simplifying design and analysis of complex predicate encryption schemes. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 627–656. Springer, Heidelberg, May 2017.
[6] J. A. Akinyele, C. Garman, and S. Hohenberger. Automating fast and secure translations from type-I to type-III pairing schemes. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 1370–1381. ACM Press, Oct. 2015.
[7] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, pages 111–128, 2013.
[8] J. A. Akinyele, M. Green, and S. Hohenberger. Using SMT solvers to automate design tasks for encryption and signature schemes. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 13*, pages 399–410. ACM Press, Nov. 2013.
[9] J. A. Akinyele, M. W. Pagano, M. D. Green, C. U. Lehmann, Z. N. J. Peterson, and A. D. Rubin. Securing electronic medical records using attribute-based encryption on mobile devices. In *SPSM '11*, pages 75–86, 2011.
[10] N. Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 557–577. Springer, Heidelberg, May 2014.
[11] N. Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 591–623. Springer, Heidelberg, Dec. 2016.
[12] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: an online social network with user-defined privacy. In *ACM SIGCOMM 2009*, pages 135–146, 2009.
[13] A. Beimel. Secret-sharing schemes: A survey. In *Coding and Cryptology*, pages 11–46. 2011.
[14] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.
[15] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, Heidelberg, May 1995.
[16] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Press, May 2007.
[17] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, Aug. 2004.
[18] J. Camenisch, M. Dubovitskaya, R. R. Enderlein, and G. Neven. Oblivious transfer with hidden access control from attribute-based encryption. In I. Visconti and R. D. Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 559–579. Springer, Heidelberg, Sept. 2012.
[19] J. Chen, R. Gay, and H. Wee. Improved dual system ABE in prime-order groups via predicate encodings. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, Heidelberg, Apr. 2015.
[20] J. Chen, R. Gay, and H. Wee. Improved dual system ABE in prime-order groups via predicate encodings. Cryptology ePrint Archive, Report 2015/409, 2015. http://eprint.iacr.org/2015/409.
[21] J. Chen and H. Wee. Fully, (almost) tightly secure IBE and dual system groups. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 435–460. Springer, Heidelberg, Aug. 2013.
[22] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, Aug. 2013.
[23] D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61. Springer, Heidelberg, May 2010.
[24] S. Galbraith. New discrete logarithm records, and the death of type 1 pairings. https://ellipticnews.wordpress.com/2014/02/01/new-discrete-logarithm-records-and-the-death-of-type-1-pairings/, 2014.

[25] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, pages 3113 – 3121, 2008.

[26] E. Ghadafi, N. P. Smart, and B. Warinschi. Groth-Sahai proofs revisited. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 177–192. Springer, Heidelberg, May 2010.

[27] V. Goyal, A. Jain, O. Pandey, and A. Sahai. Bounded ciphertext policy attribute based encryption. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 579–591. Springer, Heidelberg, July 2008.

[28] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 89–98. ACM Press, Oct. / Nov. 2006. Available as Cryptology ePrint Archive Report 2006/309.

[29] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. Cryptology ePrint Archive, Report 2006/309, 2006. http://eprint.iacr.org/2006/309.

[30] M. Green and S. Hohenberger. Universally composable adaptive oblivious transfer. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 179–197. Springer, Heidelberg, Dec. 2008.

[31] M. Green, S. Hohenberger, and B. Waters. Outsourcing the decryption of abe ciphertexts. In *USENIX Security Symposium 2011*, pages 34–34, 2011.

[32] M. D. Green and I. Miers. Forward secure asynchronous messaging from puncturable encryption. In *2015 IEEE Symposium on Security and Privacy*, pages 305–320. IEEE Computer Society Press, May 2015.

[33] A. Guillevic. Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In M. J. Jacobson Jr., M. E. Locasto, P. Mohassel, and R. Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 357–372. Springer, Heidelberg, June 2013.

[34] A. Guillevic, F. Morain, and E. Thomé. Solving discrete logarithms on a 170-bit MNT curve by pairing reduction. Cryptology ePrint Archive, Report 2016/507, 2016. http://eprint.iacr.org/2016/507.

[35] G. Herold, J. Hesse, D. Hofheinz, C. Ràfols, and A. Rupp. Polynomial spaces: A new framework for composite-to-prime-order transformations. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 261–279. Springer, Heidelberg, Aug. 2014.

[36] S. Hohenberger and B. Waters. Attribute-based encryption with fast decryption. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 162–179. Springer, Heidelberg, Feb. / Mar. 2013.

[37] A. Joux. *SAC 2013*, chapter A New Index Calculus Algorithm with Complexity $$L(1/4+o(1))$$ in Small Characteristic, pages 355–379. 2014.

[38] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, Apr. 2008.

[39] T. Kim and R. Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 543–571. Springer, Heidelberg, Aug. 2016.

[40] L. Kowalczyk and A. B. Lewko. Bilinear entropy expansion from the decisional linear assumption. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 524–541. Springer, Heidelberg, Aug. 2015.

[41] A. B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 318–335. Springer, Heidelberg, Apr. 2012.

[42] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May 2010.

[43] A. B. Lewko, A. Sahai, and B. Waters. Revocation systems with very small private keys. In *2010 IEEE Symposium on Security and Privacy*, pages 273–285. IEEE Computer Society Press, May 2010.

[44] A. B. Lewko and B. Waters. Unbounded HIBE and attribute-based encryption. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, Heidelberg, May 2011.

[45] B. Libert, T. Peters, M. Joye, and M. Yung. Compactly hiding linear spans - tightly secure constant-size simulation-sound QA-NIZK proofs and applications. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 681–707. Springer, Heidelberg, Nov. / Dec. 2015.

[46] T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, Aug. 2010.

[47] T. Okamoto and K. Takashima. Fully secure unbounded inner-product and attribute-based encryption. In X. Wang and K. Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 349–366. Springer, Heidelberg, Dec. 2012.

[48] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM CCS 07*, pages 195–203. ACM Press, Oct. 2007.

[49] B. Parno, M. Raykova, and V. Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 422–439. Springer, Heidelberg, Mar. 2012.

[50] Y. Rouselakis and B. Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 13*, pages 463–474. ACM Press, Nov. 2013.

[51] Y. Rouselakis and B. Waters. Efficient statically-secure large-universe multi-authority attribute-based encryption. In R. Böhme and T. Okamoto, editors, *FC 2015*, volume 8975 of *LNCS*, pages 315–332. Springer, Heidelberg, Jan. 2015.

[52] A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

[53] N. Santos, R. Rodrigues, K. P. Gummadi, and S. Saroiu. Policy-sealed data: A new abstraction for building trusted cloud services. In *USENIX Security Symposium 2012*, pages 175–188, 2012.

[54] H. Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. http://eprint.iacr.org/2007/074.

[55] P. Traynor, K. R. B. Butler, W. Enck, and P. McDaniel. Realizing massive-scale conditional access systems through attribute-based cryptosystems. In *NDSS 2008*. The Internet Society, Feb. 2008.

[56] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, Aug. 2009.

[57] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 53–70. Springer, Heidelberg, Mar. 2011.

[58] H. Wee. Dual system encryption via predicate encodings. In Y. Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 616–637. Springer, Heidelberg, Feb. 2014.

[59] B. Yang, K. Yang, Y. Qin, Z. Zhang, and D. Feng. *DAA-TZ: An Efficient DAA Scheme for Mobile Devices Using ARM TrustZone*, pages 209–227. Springer International Publishing, Cham, 2015.

A CP-ABE CORRECTNESS

We show that when S satisfies $(\mathbf{M}, \pi)$, decryption recovers the correct message with probability 1. For $\ell = 1, 2, 3$,

$$\begin{aligned}
\prod_{i\in I} \mathrm{ct}_{i,\ell}^{\gamma_i} &= \prod_{i\in I}\Big(\mathcal{H}(\pi(i)\ell 1)^{\gamma_i s_1}\cdot \mathcal{H}(\pi(i)\ell 2)^{\gamma_i s_2}\cdot \\
&\qquad \prod_{j=1}^{n_2}\left[\mathcal{H}(0j\ell 1)^{s_1}\cdot\mathcal{H}(0j\ell 2)^{s_2}\right]^{\gamma_i(\mathbf{M})_{i,j}}\Big) \\
&= \left(\prod_{j=1}^{n_2}\left[\mathcal{H}(0j\ell 1)^{s_1}\cdot\mathcal{H}(0j\ell 2)^{s_2}\right]^{\sum_{i\in I}\gamma_i(\mathbf{M})_{i,j}}\right)\cdot \\
&\qquad \left(\prod_{i\in I}\mathcal{H}(\pi(i)\ell 1)^{\gamma_i s_1}\cdot\mathcal{H}(\pi(i)\ell 2)^{\gamma_i s_2}\right) \\
&= \mathcal{H}(01\ell 1)^{s_1}\cdot\mathcal{H}(01\ell 2)^{s_2}\cdot \\
&\qquad \prod_{i\in I}\mathcal{H}(\pi(i)\ell 1)^{\gamma_i s_1}\cdot\mathcal{H}(\pi(i)\ell 2)^{\gamma_i s_2},
\end{aligned}$$

where the third equality follows from (2.1).

Now, the product of all but the first term in num is given by

$$\begin{aligned}
&\prod_{t\in\{1,2\}}\Big[e(\mathcal{H}(011t),h)^{b_1 r_1 s_t}\cdot e(\mathcal{H}(012t),h)^{b_2 r_2 s_t} \\
&\quad\cdot e(\mathcal{H}(013t),h)^{(r_1+r_2)s_t}\cdot\prod_{i\in I}\Big(e(\mathcal{H}(\pi(i)1t)^{\gamma_i},h)^{b_1 r_1 s_t} \\
&\quad\cdot e(\mathcal{H}(\pi(i)2t)^{\gamma_i},h)^{b_2 r_2 s_t}\cdot e(\mathcal{H}(\pi(i)3t)^{\gamma_i},h)^{(r_1+r_2)s_t}\Big)\Big]
\end{aligned}$$

When the above product is divided by dem, it is easy to see that we are only left with the inverse of (rest of the terms cancel out)

$$\left(\prod_{t\in\{1,2\}} e\left(g^{d_t}\cdot g^{\frac{\sigma'}{a_t}}\cdot\prod_{i\in I} g^{\frac{\gamma_i\sigma_{\pi(i)}}{a_t}}, h^{a_t s_t}\right)\right)\cdot$$
$$e\left(g^{d_3}\cdot g^{-\sigma'}\cdot\prod_{i\in I} g^{-\gamma_i\sigma_{\pi(i)}}, h^{s_1+s_2}\right),$$

which is just equal to $e(g,h)^{d_1a_1s_1+d_2a_2s_2+d_3(s_1+s_2)}$. Hence, msg is successfully recovered.

B KP-ABE SCHEME

The scheme is formally described in Figure B.1. Correctness and security of this scheme can be proved in a manner very similar to that of Fame. Note that unlike CGW's KP-ABE scheme [20, Appendix B.1], our scheme does not put an a-priori bound on the number of columns in the MSP.

C PROOF OF MAIN THEOREM

C.1 Description of hybrids

Hyb_1 has already been discussed at length in §4.1. We now provide a formal description of the rest of the hybrids, and then prove the security of Fame. Before that, it would be useful to give names to the various forms of ciphertext and keys that will be used. A key can be in one of the following forms:

- Normal: Generated in Hyb_1.
- P-normal: $\mathbf{Br}$ replaced by $\mathbf{Br}+\hat{r}\mathbf{a}^\perp$ in a Normal key, where $\hat{r}\leftarrow_R\mathbb{Z}_p$.
- P-normal*: $\sigma_y\mathbf{a}^\perp$ for all $y\in S$ and $\sigma'\mathbf{a}^\perp$ removed from a P-normal key.
- Normal*: $\mathbf{Br}+\hat{r}\mathbf{a}^\perp$ replaced by $\mathbf{Br}$ in a P-normal* key.
- P-SF*: $\alpha\mathbf{a}^\perp$ added to the last component (sk$'$) of a P-normal* key, where $\alpha\leftarrow_R\mathbb{Z}_p$.
- SF*: $\mathbf{Br}+\hat{r}\mathbf{a}^\perp$ replaced by $\mathbf{Br}$ in a P-SF* key.

A ciphertext can be either:

- Normal*: Generated in Hyb_1.
- SF*: $\mathbf{As}$ replaced by $\mathbf{As}+\hat{s}\mathbf{b}^\perp$ in a Normal* ciphertext, where $\hat{s}\leftarrow_R\mathbb{Z}_p$.
- Rnd*: msg$_b$ replaced by msg*, where msg$^\star\leftarrow_R\mathbb{G}_T$.

P and SF stand for pseudo and semi-functional, respectively, following the terminology in previous work [19, 21, 42, 56].

The first objective of our proof is to remove the extra $\sigma_y\mathbf{a}^\perp$ and $\sigma'\mathbf{a}^\perp$ components from all the keys. To do this, we change the form of the very first key from Normal to P-normal in $\mathsf{Hyb}_{2,1,1}$, then change it to P-normal* in $\mathsf{Hyb}_{2,2,1}$, and finally to Normal* in $\mathsf{Hyb}_{2,3,1}$. We then carry out the same steps for the second key, third key, and so on, until all the keys are of type Normal*. Thus, we define the following hybrids for $q=1,\dots,Q$, where Q is the total number of key queries $\mathcal{A}$ makes.

- $\underline{\mathsf{Hyb}_{2,1,q}}$: Same as Hyb_1 except first $i-1$ keys are Normal*, ith key is P-normal, and rest are Normal.
- $\underline{\mathsf{Hyb}_{2,2,q}}$: Same as $\mathsf{Hyb}_{2,1,q}$ except ith key is P-normal*.
- $\underline{\mathsf{Hyb}_{2,3,q}}$: Same as $\mathsf{Hyb}_{2,2,q}$ except ith key is Normal*.

The next objective is to show that the challenge ciphertext is able to hide the message encrypted if none of the keys issued can decrypt it individually. Here we first change the form of ciphertext from Normal* to SF* in Hyb_3. Then one by one we change all the keys from Normal* to P-normal*, then to P-SF*, and finally to SF*. The extra component $\alpha\mathbf{a}^\perp$ now present in all the keys helps us to then make the ciphertext Rnd*. Thus, the hybrids are

- $\underline{\mathsf{Hyb}_3}$: Same as $\mathsf{Hyb}_{2,3,Q}$ except ciphertext is SF*.
- $\underline{\mathsf{Hyb}_{4,1,q}}$: Same as Hyb_3 except first $i-1$ keys are SF*, ith key is P-normal*, and rest are Normal*.
- $\underline{\mathsf{Hyb}_{4,2,q}}$: Same as $\mathsf{Hyb}_{4,1,q}$ except ith key is P-SF*.
- $\underline{\mathsf{Hyb}_{4,3,q}}$: Same as $\mathsf{Hyb}_{4,2,q}$ except ith key is SF*.
- $\underline{\mathsf{Hyb}_5}$: Same as $\mathsf{Hyb}_{4,3,Q}$ except ciphertext is Rnd*.

Note that in all the hybrids, the random oracle is simulated in the same way as in Hyb_1. Also, two additional hybrids $\mathsf{Hyb}_{2,3,0}$ and $\mathsf{Hyb}_{4,3,0}$ are defined to be same as Hyb_1 and Hyb_3, respectively.

C.2 Sampling algorithm

On input a prime p, recall that Samp outputs

$$\mathbf{Z}:=\begin{bmatrix}u_1 & 0\\ 0 & u_2\\ 1 & 1\end{bmatrix},\qquad \mathbf{z}^\perp:=\begin{bmatrix}u_1{}^{-1}\\ u_2{}^{-1}\\ -1\end{bmatrix},\qquad \text{(C.1)}$$

where $u_1,u_2\leftarrow_R\mathbb{Z}_p^*$. If $[\mathbf{X}||\mathbf{Y}]$ is used to denote the column-wise join of two matrices $\mathbf{X}$ and $\mathbf{Y}$, then note that $[\mathbf{Z}||\mathbf{z}^\perp]$ is a full-rank matrix. Also, observe that the matrix $\mathbf{Z}$ here has exactly the same distribution as $\mathbf{A}$ from the DLIN assumption, and that $\mathbf{Z}^\top\mathbf{z}^\perp=\mathbf{0}$. We will need the following *basis* lemma from [19].

Lemma C.1 (Basis lemma). *Let $(\mathbf{Z}_1,\mathbf{z}_1^\perp)$ and $(\mathbf{Z}_2,\mathbf{z}_2^\perp)$ be two independent samples drawn from Samp(p). Then with probability $1-1/p$, it holds that $[\mathbf{Z}_1||\mathbf{z}_2^\perp]$ and $[\mathbf{Z}_2||\mathbf{z}_1^\perp]$ are full-rank matrices as well as $\langle\mathbf{z}_1^\perp,\mathbf{z}_2^\perp\rangle\neq 0$.*

C.3 Indistinguishability of hybrids

In the following, $\mathsf{Adv}_{i,j}^{\mathcal{A}}(\lambda)$ denotes the advantage of an adversary $\mathcal{A}$ in distinguishing Hyb_i from Hyb_j when the security parameter is λ. Although the indistinguishability of every pair of hybrids below holds irrespective of the value of bit b given to the challenger, we do not put this explicitly into the theorem statements.

Lemma C.2. *For any adversary $\mathcal{A}$, $\mathsf{Adv}_{0,1}^{\mathcal{A}}(\lambda)=0$.*

Proof. First of all, it is easy to see that the master public and secret keys are generated identically in both the hybrids because the first output of Samp has exactly the same distribution as $\mathbf{A}$ from the DLIN assumption (§2.5). Further, the response of Chal on an oracle query of the form $x\ell t$ in Hyb_1 is $[(\mathbf{W}_x^\top\mathbf{A})_{\ell,t}]_1$, whose exponent is $a_t(\mathbf{W}_x)_{t,\ell}+(\mathbf{W}_x)_{3,\ell}$, for randomly chosen $(\mathbf{W}_x)_{t,\ell}$ and $(\mathbf{W}_x)_{3,\ell}$. Hence, $[(\mathbf{W}_x^\top\mathbf{A})_{\ell,t}]_1$ is independently and uniformly distributed for every x, ℓ, t. In the same way, we can argue that the response to queries of the form $0j\ell t$ are also independent and uniform over $\mathbb{G}$. Thus, Chal perfectly simulates a random oracle.

If we implicitly set the responses of random oracle in Hyb_0 to be the ones generated by Chal in Hyb_1, then the ct$_{i,\ell}$ component

- <u>Setup(1^λ)</u> Same Setup as that of FAME.
- <u>KeyGen(msk, $(\mathbf{M}, \pi)$)</u> Pick $r_1, r_2 \leftarrow_R \mathbb{Z}_p$ and compute
$$\mathsf{sk}_0 := (h^{b_1 r_1}, h^{b_2 r_2}, h^{r_1+r_2})$$
using h, b_1, b_2 from msk. Pick $\sigma'_2, \dots, \sigma'_{n_2} \leftarrow_R \mathbb{Z}_p$. For all $i = 1, \dots, n_1$ and $t = 1, 2$, compute
$$\mathsf{sk}_{i,t} := \mathcal{H}(\pi(i)1t)^{\frac{b_1 r_1}{a_t}} \cdot \mathcal{H}(\pi(i)2t)^{\frac{b_2 r_2}{a_t}} \cdot \mathcal{H}(\pi(i)3t)^{\frac{r_1+r_2}{a_t}} \cdot g^{\frac{\sigma_i}{a_t}} \cdot \left(g^{d_t}\right)^{(\mathbf{M})_{i,1}} \cdot \prod_{j=2}^{n_2} \left[\mathcal{H}(0j1t)^{\frac{b_1 r_1}{a_t}} \cdot \mathcal{H}(0j2t)^{\frac{b_2 r_2}{a_t}} \cdot \mathcal{H}(0j3t)^{\frac{r_1+r_2}{a_t}} \cdot g^{\frac{\sigma'_j}{a_t}}\right]^{(\mathbf{M})_{i,j}},$$
$$\mathsf{sk}_{i,3} := g^{-\sigma_i} \cdot \left(g^{d_3}\right)^{(\mathbf{M})_{i,1}} \cdot \prod_{j=2}^{n_2} \left(g^{-\sigma'_j}\right)^{(\mathbf{M})_{i,j}},$$
where $\sigma_i \leftarrow_R \mathbb{Z}_p$. Set $\mathsf{sk}_i := (\mathsf{sk}_{i,1}, \mathsf{sk}_{i,2}, \mathsf{sk}_{i,3})$. Output $(\mathsf{sk}_0, \mathsf{sk}_1, \dots, \mathsf{sk}_{n_1})$ as the key.
- <u>Encrypt(pk, S, msg)</u> Pick $s_1, s_2 \leftarrow_R \mathbb{Z}_p$ and compute
$$\mathsf{ct}_0 := (H_1^{s_1}, H_2^{s_2}, h^{s_1+s_2}).$$
using pk. For all $y \in S$ and $\ell = 1, 2, 3$, compute
$$\mathsf{ct}_{y,\ell} := \mathcal{H}(y\ell 1)^{s_1} \cdot \mathcal{H}(y\ell 2)^{s_2}.$$
Set $\mathsf{ct}_y := (\mathsf{ct}_{y,1}, \mathsf{ct}_{y,2}, \mathsf{ct}_{y,3})$. Also, compute
$$\mathsf{ct}' := T_1^{s_1} \cdot T_2^{s_2} \cdot \mathsf{msg}.$$
Output $(\mathsf{ct}_0, \{\mathsf{ct}_y\}_{y \in S}, \mathsf{ct}')$ as the ciphertext.
- <u>Decrypt(pk, ct, sk)</u> Same as the decryption algorithm of FAME except that for any $i \in I$, $\mathsf{ct}_{\pi(i)}$ is used to compute num and sk_i to compute dec. Also, note that there is no sk$'$ component in the key.

Figure B.1: Key-policy attribute-based encryption.

of the challenge ciphertext in Hyb_0 is set to
$$\Big[(\mathbf{W}_{\pi(i)}^{\mathsf{T}}\mathbf{A})_{\ell,1}s_1 + (\mathbf{W}_{\pi(i)}^{\mathsf{T}}\mathbf{A})_{\ell,2}s_2 + \sum_j \left\{(\mathbf{U}_j^{\mathsf{T}}\mathbf{A})_{\ell,1}s_1 + (\mathbf{U}_j^{\mathsf{T}}\mathbf{A})_{\ell,2}s_2\right\}(\mathbf{M})_{i,j}\Big]_1$$
for $\ell \in \{1, 2, 3\}$. Therefore, ct_i is equal to
$$[\mathbf{W}_{\pi(i)}^{\mathsf{T}}\mathbf{As} + (\mathbf{M})_{i,1}\mathbf{U}_1^{\mathsf{T}}\mathbf{As} + \ldots + (\mathbf{M})_{i,n_2}\mathbf{U}_{n_2}^{\mathsf{T}}\mathbf{As}]_1,$$
if $\mathbf{s}$ is defined to be $(s_1, s_2)^{\mathsf{T}}$. We can also rewrite ct_0 and ct' as $[\mathbf{As}]_2$ and $[\mathbf{d}^{\mathsf{T}}\mathbf{As}]_T \cdot \mathsf{msg}_b$, respectively. Thus, we obtain a ciphertext identical to the one in Hyb_1.

Let us now turn to the key component $\mathsf{sk}_{y,t}$, which is implicitly set to
$$\Big[(\mathbf{W}_y^{\mathsf{T}}\mathbf{A})_{1,t}\frac{b_1 r_1}{a_t} + (\mathbf{W}_y^{\mathsf{T}}\mathbf{A})_{2,t}\frac{b_2 r_2}{a_t} + (\mathbf{W}_y^{\mathsf{T}}\mathbf{A})_{3,t}\frac{r_1 + r_2}{a_t} + \frac{\sigma_y}{a_t}\Big]_1$$
for $t \in \{1, 2\}$. If we denote the (i, j)th element of $\mathbf{W}_y$ by $w_{i,j}$, then the exponent of g in $\mathsf{sk}_{y,t}$ can be expanded as:
$$\begin{aligned}&(a_t w_{t,1} + w_{3,1})\frac{b_1 r_1}{a_t} + (a_t w_{t,2} + w_{3,2})\frac{b_2 r_2}{a_t} + (a_t w_{t,3} + w_{3,3})\frac{r_1 + r_2}{a_t} + \frac{\sigma_y}{a_t}\\ &= (w_{t,1}b_1 + w_{t,3})r_1 + (w_{t,2}b_2 + w_{t,3})r_2 + \frac{1}{a_t}\left[(w_{3,1}b_1 + w_{3,3})r_1 + (w_{3,2}b_2 + w_{3,3})r_2 + \sigma_y\right]\\ &= (\mathbf{W}_y\mathbf{Br})_t + a_t^{-1}\left[(\mathbf{W}_y\mathbf{Br})_3 + \sigma_y\right],\end{aligned}$$
where $\mathbf{r} := (r_1, r_2)^{\mathsf{T}}$. The third part of sk_y is $g^{-\sigma_y}$, whose exponent can be written as $(\mathbf{W}_y\mathbf{Br})_3 - \left[(\mathbf{W}_y\mathbf{Br})_3 + \sigma_y\right]$. Now note that if σ_y is uniformly random, then so is $(\mathbf{W}_y\mathbf{Br})_3 + \sigma_y$. Hence, sk_y is identically distributed to $[\mathbf{W}_y\mathbf{Br} + \sigma_y\mathbf{a}^{\perp}]_1$.

In the same way, we can show that sk$'$ is identically distributed to $[\mathbf{d} + \mathbf{U}_1\mathbf{Br} + \sigma'\mathbf{a}^{\perp}]_1$ for a randomly chosen σ'. Finally, sk_0 can be described succinctly as $[\mathbf{Br}]_2$. Thus, we obtain a key identical to the one output in Hyb_1. □

LEMMA C.3. *For all $q = 1, \dots, Q$ and* PPT *adversaries $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{B}$ such that*
$$\mathsf{Adv}^{\mathcal{A}}_{(2,3,q-1),(2,1,q)}(\lambda) \leq \mathsf{Adv}^{\mathcal{B}}_{\mathsf{DLIN}}(\lambda) + 1/p.$$

PROOF. The only difference between $\mathsf{Hyb}_{2,3,q-1}$ and $\mathsf{Hyb}_{2,1,q}$ is in the form of the ith key issued by the challenger. In the former case, this key is Normal while in the latter, it is P-normal. We design an adversary $\mathcal{B}$ that converts any advantage $\mathcal{A}$ has in distinguishing the two hybrids into an (almost) equal advantage in breaking the DLIN assumption.

$\mathcal{B}$ gets $([\mathbf{B}]_1, [\mathbf{B}]_2, [\mathbf{Br}^*]_1, [\mathbf{Br}^*]_2)$ or $([\mathbf{B}]_1, [\mathbf{B}]_2, [\mathbf{r}']_1, [\mathbf{r}']_2)$ as the DLIN challenge, and simulates the challenger in the IND-CPA security game that it plays with $\mathcal{A}$. It draws $(\mathbf{A}, \mathbf{a}^\perp)$ from Samp and $\mathbf{d} \leftarrow_R \mathbb{Z}_p^3$, and gives $([\mathbf{A}]_2, [\mathbf{d}^\mathsf{T}\mathbf{A}]_T)$ to $\mathcal{A}$ as the public key. Further, it simulates the random oracle in the same way as the challenger does in $\mathsf{Hyb}_{2,3,q-1}$ or $\mathsf{Hyb}_{2,1,q}$.

Since $[\mathbf{B}||\mathbf{a}^\perp]$ is a full-rank matrix (except with probability $1/p$, see Lem C.1), we can say that $\mathcal{B}$ receives $([\mathbf{B}]_1, [\mathbf{B}]_2, [\mathbf{Br}^* + \hat{r}\mathbf{a}^\perp]_1, [\mathbf{Br}^* + \hat{r}\mathbf{a}^\perp]_2)$ as the DLIN tuple, where $\hat{r}$ is either zero or a randomly chosen value from $\mathbb{Z}_p$.

It is straightforward for $\mathcal{B}$ to generate the challenge ciphertext. To generate any of the first $i-1$ keys, $\mathcal{B}$ picks $\mathbf{r} \leftarrow_R \mathbb{Z}_p^2$ and outputs $([\mathbf{Br}]_2, \{[\mathbf{W}_y\mathbf{Br}]_1\}_{y\in S}, [\mathbf{d} + \mathbf{U}_1\mathbf{Br}]_1)$ [14]; only $[\mathbf{B}]_1, [\mathbf{B}]_2$ are required for this. The other keys, except the ith, are also easily generated since $\mathcal{B}$ knows $\mathbf{a}^\perp$.

Now, in order to generate the ith key, $\mathcal{B}$ picks $\sigma_y \leftarrow_R \mathbb{Z}_p$ for $y \in S$ and $\sigma' \leftarrow_R \mathbb{Z}_p$, and outputs

$$([\mathbf{Br}^* + \hat{r}\mathbf{a}^\perp]_2, \{[\mathbf{W}_y(\mathbf{Br}^* + \hat{r}\mathbf{a}^\perp) + \sigma_y\mathbf{a}^\perp]_1\}_{y\in S}, [\mathbf{d} + \mathbf{U}_1(\mathbf{Br}^* + \hat{r}\mathbf{a}^\perp) + \sigma'\mathbf{a}^\perp]_1).$$

It is easy to see that if $\hat{r} = 0$, the view of $\mathcal{A}$ is identical to that in $\mathsf{Hyb}_{2,3,q-1}$; otherwise, the view is identical to $\mathsf{Hyb}_{2,1,q}$. □

Lemma C.4. *For all $q = 1, \ldots, Q$ and adversaries $\mathcal{A}$,*

$$\mathsf{Adv}^{\mathcal{A}}_{(2,1,q),(2,2,q)}(\lambda) \le 2/p.$$

Proof. We want to prove that the view of any adversary (even unbounded) in $\mathsf{Hyb}_{2,1,q}$ is identically distributed to its view in $\mathsf{Hyb}_{2,2,q}$ (except with negligible probability). Towards this, let $\mathbf{V}$ be a matrix defined by the product of $\mathbf{a}^\perp$ with the transpose of $\mathbf{b}^\perp$. Note that $\mathbf{V}^\mathsf{T}\mathbf{A} = \mathbf{VB} = 0$ and $\mathbf{Va}^\perp = (\mathbf{a}^\perp{\mathbf{b}^\perp}^\mathsf{T})\mathbf{a}^\perp = \mathbf{a}^\perp({\mathbf{b}^\perp}^\mathsf{T}\mathbf{a}^\perp) = ({\mathbf{a}^\perp}^\mathsf{T}\mathbf{b}^\perp)\mathbf{a}^\perp$ since ${\mathbf{b}^\perp}^\mathsf{T}\mathbf{a}^\perp$ is nothing but the inner product of $\mathbf{a}^\perp$ and $\mathbf{b}^\perp$. Let β denote this inner product, which is non-zero except with probability $1/p$ (see Lem C.1).

Consider the hybrid $\mathsf{Hyb}_{2,1,q}$. Suppose $\mathbf{W}_x$ is implicitly set to $\mathbf{W}_x^* := \mathbf{W}_x - \sigma_x(\beta\hat{r})^{-1}\mathbf{V}$ and $\mathbf{U}_j$ to $\mathbf{U}_j^* := \mathbf{U}_j - \sigma'(\beta\hat{r})^{-1}\mathbf{V}$, where $\sigma_x, \sigma', \hat{r} \leftarrow_R \mathbb{Z}_p$ ($\hat{r} \neq 0$ with probability $1 - 1/p$). This does not affect the distribution of these matrices because they are chosen at random. The ciphertext is not affected either since $(\mathbf{W}^*_{\pi(i)})^\mathsf{T}\mathbf{A} = \mathbf{W}^\mathsf{T}_{\pi(i)}\mathbf{A}$ and, similarly, $(\mathbf{U}_j^*)^\mathsf{T}\mathbf{A} = \mathbf{U}_j^\mathsf{T}\mathbf{A}$. Analogously, the form of all the keys except the ith one remains unchanged. In the case of ith key, we have

$$\begin{aligned}
&\mathbf{W}_y^*(\mathbf{Br} + \hat{r}\mathbf{a}^\perp) + \sigma_y\mathbf{a}^\perp \\
&\quad = (\mathbf{W}_y - \sigma_y(\beta\hat{r})^{-1}\mathbf{V})(\mathbf{Br} + \hat{r}\mathbf{a}^\perp) + \sigma_y\mathbf{a}^\perp \\
&\quad = \mathbf{W}_y\mathbf{Br} - \sigma_y(\beta\hat{r})^{-1}\hat{r}\mathbf{Va}^\perp + \mathbf{W}_y\hat{r}\mathbf{a}^\perp + \sigma_y\mathbf{a}^\perp \\
&\quad = \mathbf{W}_y(\mathbf{Br} + \hat{r}\mathbf{a}^\perp) - \sigma_y\beta^{-1}\beta\mathbf{a}^\perp + \sigma_y\mathbf{a}^\perp \\
&\quad = \mathbf{W}_y(\mathbf{Br} + \hat{r}\mathbf{a}^\perp)
\end{aligned}$$

and, similarly, $\mathbf{d} + \mathbf{U}_1^*(\mathbf{Br} + \hat{r}\mathbf{a}^\perp) + \sigma'\mathbf{a}^\perp = \mathbf{d} + \mathbf{U}_1(\mathbf{Br} + \hat{r}\mathbf{a}^\perp)$, which is how the ith key of $\mathsf{Hyb}_{2,2,q}$ is distributed. (Recall that the hybrids under consideration in this proof only differed on the ith key.) □

Lemma C.5. *For all* PPT *adversaries $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathcal{A}}_{(2,3,Q),3}(\lambda) \le \mathsf{Adv}^{\mathcal{B}}_{\mathsf{DLIN}}(\lambda) + 1/p.$$

Proof. The only difference between $\mathsf{Hyb}_{2,3,Q}$ and Hyb_3 is in the form of the challenge ciphertext; all the keys are Normal* in both the cases. $\mathcal{B}$ gets $([\mathbf{A}]_1, [\mathbf{A}]_2, [\mathbf{As}]_1, [\mathbf{As}]_2)$ or $([\mathbf{A}]_1, [\mathbf{A}]_2, [\mathbf{s}']_1, [\mathbf{s}']_2)$ as the DLIN challenge. It draws $(\mathbf{B}, \mathbf{b}^\perp)$ from Samp and $\mathbf{d} \leftarrow_R \mathbb{Z}_p^3$, and gives $([\mathbf{A}]_2, [\mathbf{d}^\mathsf{T}\mathbf{A}]_T)$ to $\mathcal{A}$ as the public key. Using $\mathbf{B}$, it can easily generate keys for any set of attributes.

Since $[\mathbf{A}||\mathbf{b}^\perp]$ is a full-rank matrix, we can say that $\mathcal{B}$ receives $([\mathbf{A}]_1, [\mathbf{A}]_2, [\mathbf{As} + \hat{s}\mathbf{b}^\perp]_1, [\mathbf{As} + \hat{s}\mathbf{b}^\perp]_2)$ as the DLIN tuple, where $\hat{s}$ is either zero or a randomly chosen value from $\mathbb{Z}_p$. Now, when $\mathcal{A}$ sends $\mathsf{msg}_0, \mathsf{msg}_1$ and a policy $(\mathbf{M}, \pi)$, $\mathcal{B}$ outputs

$$\begin{aligned}
\mathsf{ct}_0 &:= [\mathbf{As} + \hat{s}\mathbf{b}^\perp]_2 \\
\mathsf{ct}_i &:= [\mathbf{W}^\mathsf{T}_{\pi(i)}(\mathbf{As} + \hat{s}\mathbf{b}^\perp) + \sum_{j=1}^{n_2}(\mathbf{M})_{i,j}\mathbf{U}_j^\mathsf{T}(\mathbf{As} + \hat{s}\mathbf{b}^\perp)]_1 \\
\mathsf{ct}' &:= [\mathbf{d}^\mathsf{T}(\mathbf{As} + \hat{s}\mathbf{b}^\perp)]_T \cdot \mathsf{msg}_b,
\end{aligned}$$

for $i = 1, \ldots, n_1$. It is easy to see that if $\hat{s} = 0$, then the view of $\mathcal{A}$ is identical to that in $\mathsf{Hyb}_{2,3,Q}$; otherwise, the view is identical to that in Hyb_3. (Note that $[\mathbf{A}]_1$ is needed to simulate the random oracle.) □

Lemma C.6. *For all $q = 1, \ldots, Q$ and* PPT *adversaries $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathcal{A}}_{(4,3,q-1),(4,1,q)}(\lambda) \le \mathsf{Adv}^{\mathcal{B}}_{\mathsf{DLIN}}(\lambda) + 1/p.$$

Proof. $\mathcal{B}$ draws $(\mathbf{A}, \mathbf{a}^\perp)$ from Samp and $\mathbf{d} \leftarrow_R \mathbb{Z}_p^3$, and gives $([\mathbf{A}]_2, [\mathbf{d}^\mathsf{T}\mathbf{A}]_T)$ to $\mathcal{A}$ as the public key. It also uses $\mathbf{A}$ to simulate the random oracle queries. As in Lem C.3, we can assume that $\mathcal{B}$ receives $([\mathbf{B}]_1, [\mathbf{B}]_2, [\mathbf{Br}^* + \hat{r}\mathbf{a}^\perp]_1, [\mathbf{Br}^* + \hat{r}\mathbf{a}^\perp]_2)$ as the DLIN tuple, where $\hat{r}$ is either zero or a randomly chosen value from $\mathbb{Z}_p$.

It is not immediately clear how $\mathcal{B}$ will generate the challenge ciphertext since it does not know $\mathbf{b}^\perp$. However, observe that when $\mathbf{s} \leftarrow_R \mathbb{Z}_p^2$ and $\hat{s} \leftarrow_R \mathbb{Z}_p$, $\mathbf{As} + \hat{s}\mathbf{b}^\perp$ is a uniformly distributed vector over $\mathbb{Z}_p^3$. Thus, $\mathcal{B}$ just picks a random vector $\mathbf{s}'$ from $\mathbb{Z}_p^3$ and outputs $([\mathbf{s}']_2, \left\{[\mathbf{W}^\mathsf{T}_{\pi(i)}\mathbf{s}' + \sum_j(\mathbf{M})_{i,j}\mathbf{U}_j^\mathsf{T}\mathbf{s}']_1\right\}_{i\in\{1,\ldots,n_1\}}, [\mathbf{d}^\mathsf{T}\mathbf{s}']_T \cdot \mathsf{msg}_b)$ as the ciphertext.

To generate a SF* key, $\mathcal{B}$ picks $\mathbf{r} \leftarrow_R \mathbb{Z}_p^2$ and outputs $([\mathbf{Br}]_2, \{[\mathbf{W}_y\mathbf{Br}]_1\}_{y\in S}, [\mathbf{d} + \alpha\mathbf{a}^\perp + \mathbf{U}_1\mathbf{Br}]_1)$, where $\alpha \leftarrow_R \mathbb{Z}_p$. The Normal* keys are also generated in a similar way, with the only difference being that they don't have any $\mathbf{a}^\perp$ component. Finally, $\mathcal{B}$ outputs $([\mathbf{Br}^* + \hat{r}\mathbf{a}^\perp]_2, \{[\mathbf{W}_y(\mathbf{Br}^* + \hat{r}\mathbf{a}^\perp)]_1\}_{y\in S}, [\mathbf{d} + \mathbf{U}_1(\mathbf{Br}^* + \hat{r}\mathbf{a}^\perp)]_1)$ as the ith key, using the last two terms from the assumption. It is clear that if $\hat{r} = 0$, then this key is Normal*; else it is P-normal*. □

Lemma C.7. *For all $q = 1, \ldots, Q$ and adversaries $\mathcal{A}$,*

$$\mathsf{Adv}^{\mathcal{A}}_{(4,1,q),(4,2,q)}(\lambda) \le 2/p.$$

[14] A separate $\mathbf{r}$ is used for each key.

Proof. The only difference between $\mathsf{Hyb}_{4,1,q}$ and $\mathsf{Hyb}_{4,2,q}$ is in the form of the ith key. This key is $\mathsf{P\text{-}normal}^\star$ in the former case but $\mathsf{P\text{-}SF}^\star$ in the latter. The challenge ciphertext is $\mathsf{SF}^\star$ in both the cases, the first $i-1$ keys are $\mathsf{SF}^\star$ and the last $q-i$ are keys are $\mathsf{Normal}^\star$.

First of all, like Lem C.4, let $\mathbf{V}$ be the matrix $\mathbf{a}^\perp {\mathbf{b}^\perp}^\mathsf{T}$, and recall that $\mathbf{V}^\mathsf{T}\mathbf{A} = \mathbf{V}\mathbf{B} = 0$. Also, if $\beta := \langle \mathbf{a}^\perp, \mathbf{b}^\perp \rangle$, then $\mathbf{V}\mathbf{a}^\perp = \beta\mathbf{a}^\perp$ and $\mathbf{V}^\mathsf{T}\mathbf{b}^\perp = \beta\mathbf{b}^\perp$. We will also exploit the fact that none of the keys $\mathcal{A}$ requests can decrypt the challenge ciphertext. So let $\mathbf{w} = (w_1, \ldots, w_{n_2})$ be the vector guaranteed by Lem 2.2 in this case.

Consider the hybrid $\mathsf{Hyb}_{4,1,q}$ and implicitly set $\mathbf{W}_x$ to $\mathbf{W}_x + \mu_x\beta^{-1}\mathbf{V}$ and $\mathbf{U}_j$ to $\mathbf{U}_j + \alpha w_j \beta^{-1}\mathbf{V}$, where $\mu_x, \alpha \leftarrow_R \mathbb{Z}_p$. The exponent of ct_i then becomes

$$
\begin{aligned}
&(\mathbf{W}_{\pi(i)} + \mu_{\pi(i)}\beta^{-1}\mathbf{V})^\mathsf{T}(\mathbf{A}\mathbf{s} + \hat{s}\mathbf{b}^\perp) + \\
&\qquad \sum_{j=1}^{n_2} (\mathbf{M})_{i,j}(\mathbf{U}_j + \alpha w_j\beta^{-1}\mathbf{V})^\mathsf{T}(\mathbf{A}\mathbf{s} + \hat{s}\mathbf{b}^\perp) \\
&= \mathbf{W}_{\pi(i)}^\mathsf{T}(\mathbf{A}\mathbf{s} + \hat{s}\mathbf{b}^\perp) + \mu_{\pi(i)}\hat{s}\mathbf{b}^\perp + \sum_j (\mathbf{M})_{i,j}\mathbf{U}_j^\mathsf{T}(\mathbf{A}\mathbf{s} + \hat{s}\mathbf{b}^\perp) \\
&\qquad + \left(\alpha\hat{s}\sum_j (\mathbf{M})_{i,j} w_j\right)\mathbf{b}^\perp \\
&= \mathbf{W}_{\pi(i)}^\mathsf{T}(\mathbf{A}\mathbf{s} + \hat{s}\mathbf{b}^\perp) + \sum_j (\mathbf{M})_{i,j}\mathbf{U}_j^\mathsf{T}(\mathbf{A}\mathbf{s} + \hat{s}\mathbf{b}^\perp) \\
&\qquad + \left(\mu_{\pi(i)} + \alpha\sum_j (\mathbf{M})_{i,j} w_j\right)\hat{s}\mathbf{b}^\perp. \qquad \text{(C.2)}
\end{aligned}
$$

The exponent of sk_y in the ith key is now given by

$$(\mathbf{W}_y + \mu_y\beta^{-1}\mathbf{V})(\mathbf{B}\mathbf{r} + \hat{r}\mathbf{a}^\perp) = \mathbf{W}_y(\mathbf{B}\mathbf{r} + \hat{r}\mathbf{a}^\perp) + \mu_y\hat{r}\mathbf{a}^\perp,$$

and that of sk' is given by

$$
\begin{aligned}
&\mathbf{d} + (\mathbf{U}_1 + \alpha w_1\beta^{-1}\mathbf{V})(\mathbf{B}\mathbf{r} + \hat{r}\mathbf{a}^\perp) \\
&= \mathbf{d} + \mathbf{U}_1(\mathbf{B}\mathbf{r} + \hat{r}\mathbf{a}^\perp) + \alpha w_1\hat{r}\mathbf{a}^\perp.
\end{aligned}
$$

We do not need to look at other components of the ciphertext or the ith key because they do have any term involving $\mathbf{W}_x$ or $\mathbf{U}_j$. Further, any other key is not affected since the terms added to $\mathbf{W}_x$ and $\mathbf{U}_j$ are orthogonal to $\mathbf{B}$.

For an $i \in \{1, \ldots, n_1\}$, we have two possibilities. If $\pi(i) \in S$, then we know that $\sum(\mathbf{M})_{i,j}w_j = 0$. Else, none of the key components have $\mu_{\pi(i)}$, or ct_i is the only place where $\mu_{\pi(i)}$ appears[15]. So for every i, we can replace $\mu_{\pi(i)} + \alpha\sum(\mathbf{M})_{i,j}w_j$ by $\mu_{\pi(i)}$ in (C.2). Further, $\alpha w_1\hat{r}\mathbf{a}^\perp$ in sk' could be replaced by $\alpha\mathbf{a}^\perp$ without affecting the distribution as it is the only term in the adversary's view that depends on α now and $w_1\hat{r} \neq 0$ (provided $\hat{r} \neq 0$, which occurs with probability $1 - 1/p$).

[15]This is where we need π to be an injective function. If two or more rows map to the same attribute, then the argument breaks down.

After making the changes described above, we have

$$
\begin{aligned}
\mathsf{ct}_i &= [\mathbf{W}_{\pi(i)}^\mathsf{T}(\mathbf{A}\mathbf{s} + \hat{s}\mathbf{b}^\perp) + \mu_{\pi(i)}\hat{s}\mathbf{b}^\perp + \\
&\qquad \sum_j (\mathbf{M})_{i,j}\mathbf{U}_j^\mathsf{T}(\mathbf{A}\mathbf{s} + \hat{s}\mathbf{b}^\perp)]_1 \\
\mathsf{sk}_y &= [\mathbf{W}_y(\mathbf{B}\mathbf{r} + \hat{r}\mathbf{a}^\perp) + \mu_y\hat{r}\mathbf{a}^\perp]_1 \\
\mathsf{sk}' &= [\mathbf{d} + \mathbf{U}_1(\mathbf{B}\mathbf{r} + \hat{r}\mathbf{a}^\perp) + \alpha\mathbf{a}^\perp]_1.
\end{aligned}
$$

It is now easy to show that if we just replace $\mathbf{W}_x$ with $\mathbf{W}_x - \mu_x\beta^{-1}\mathbf{V}$, then the challenge ciphertext becomes $\mathsf{SF}^\star$ once again, the ith key becomes $\mathsf{P\text{-}SF}^\star$ as desired, and rest of the keys are not affected like before. □

Lemma C.8. *For all adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}_{(4,3,Q),5}(\lambda) \leq 2/p$.*

Proof. The only difference between $\mathsf{Hyb}_{4,3,Q}$ and Hyb_5 is that the ciphertext in $\mathsf{Hyb}_{4,3,Q}$ is an encryption of msg_b, while it is an encryption of a random message in Hyb_5. So suppose we implicitly set $\mathbf{d}$ chosen during the set-up process of $\mathsf{Hyb}_{4,3,Q}$ to $\mathbf{d} - \delta\mathbf{a}^\perp$, for $\delta \leftarrow_R \mathbb{Z}_p$. There are only three places where $\mathbf{d}$ appears in the view of an adversary: in the public key, the last component of all the keys, and the last component of challenge ciphertext. Among them, the public key is clearly not affected since $(\mathbf{d} - \delta\mathbf{a}^\perp)^\mathsf{T}\mathbf{A} = \mathbf{d}^\mathsf{T}\mathbf{A}$. All the $\mathsf{SF}^\star$ keys are not affected either because $(\mathbf{d} - \delta\mathbf{a}^\perp) + \mathbf{U}_1\mathbf{B}\mathbf{r} + \alpha\mathbf{a}^\perp = \mathbf{d} + \mathbf{U}_1\mathbf{B}\mathbf{r} + (\delta + \alpha)\mathbf{a}^\perp$, which is identically distributed to $\mathbf{d} + \mathbf{U}_1\mathbf{B}\mathbf{r} + \alpha\mathbf{a}^\perp$ since α is a random value.

Lastly, we have $[\mathbf{d}^\mathsf{T}(\mathbf{A}\mathbf{s} + \hat{s}\mathbf{b}^\perp)]_T \cdot \mathsf{msg}_b$ as the last component of the ciphertext in $\mathsf{Hyb}_{4,3,Q}$, which now becomes

$$
\begin{aligned}
&[(\mathbf{d} - \delta\mathbf{a}^\perp)^\mathsf{T}(\mathbf{A}\mathbf{s} + \hat{s}\mathbf{b}^\perp)]_T \cdot \mathsf{msg}_b \\
&\qquad = [\mathbf{d}^\mathsf{T}(\mathbf{A}\mathbf{s} + \hat{s}\mathbf{b}^\perp) + \delta\hat{s}\langle \mathbf{a}^\perp, \mathbf{b}^\perp \rangle]_T \cdot \mathsf{msg}_b \\
&\qquad = [\mathbf{d}^\mathsf{T}(\mathbf{A}\mathbf{s} + \hat{s}\mathbf{b}^\perp)]_T \cdot e(g,h)^{\delta\hat{s}\langle \mathbf{a}^\perp, \mathbf{b}^\perp \rangle} \cdot \mathsf{msg}_b.
\end{aligned}
$$

Note that δ does not appear in any other part of the ciphertext, or in any of the keys or the master public key. Also recall that with probability $1 - 1/p$, the inner-product of $\mathbf{a}^\perp$ and $\mathbf{b}^\perp$ is not zero (see Lem C.1). Hence, if $\hat{s} \neq 0$, which happens with probability $1 - 1/p$, $\delta\hat{s}\langle \mathbf{a}^\perp, \mathbf{b}^\perp \rangle$ is uniformly distributed over $\mathbb{Z}_p$. Thus, the ciphertext is now an encryption of a random message. □

C.4 Proof of Theorem 4.1

We have shown that $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$ in Lem C.2, $\mathsf{Hyb}_{2,3,q-1} \approx \mathsf{Hyb}_{2,1,q}$ in Lem C.3, $\mathsf{Hyb}_{2,1,q} \equiv \mathsf{Hyb}_{2,2,q}$ in Lem C.4, $\mathsf{Hyb}_{2,3,Q} \approx \mathsf{Hyb}_3$ in Lem C.5, $\mathsf{Hyb}_{4,3,q-1} \approx \mathsf{Hyb}_{4,1,q}$ in Lem C.6, $\mathsf{Hyb}_{4,1,q} \equiv \mathsf{Hyb}_{4,2,q}$ in Lem C.7, and $\mathsf{Hyb}_{4,3,Q} \equiv \mathsf{Hyb}_5$ in Lem C.8, for all $q = 1, \ldots, Q$, where $\equiv$ and $\approx$ denote statistical and computational indistinguishability, respectively, from the point of view of an adversary. ($\mathsf{Hyb}_{2,3,0}$ and $\mathsf{Hyb}_{4,3,0}$ are defined to be same as Hyb_1 and Hyb_3, respectively.) We omit a proof for the indistinguishability of $\mathsf{Hyb}_{2,2,q}$ and $\mathsf{Hyb}_{2,3,q}$ because it is completely analogous to that of $\mathsf{Hyb}_{2,3,q-1}$ and $\mathsf{Hyb}_{2,1,q}$. Also, $\mathsf{Hyb}_{4,2,q} \approx \mathsf{Hyb}_{4,3,q}$ can be proved in a manner similar to $\mathsf{Hyb}_{4,3,q-1} \approx \mathsf{Hyb}_{4,1,q}$.

In fact, the hybrids are indistinguishable irrespective of the bit b given to the challenger. In other words, none of the proofs have anything to do with the value of b. Thus, Hyb_0 (main scheme) is indistinguishable from Hyb_5 whether we start from $b = 0$ or $b = 1$, proving the theorem.

D BSW CP-ABE SCHEME

Below is the version of Bethencourt et al.'s CP-ABE scheme [16] that we implemented in asymmetric groups.

- $\underline{\mathsf{Setup}(1^\lambda)}$ Run $\mathsf{GroupGen}(1^\lambda)$ to obtain $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h)$. Pick $\alpha, \beta \leftarrow_R \mathbb{Z}_p$. Output $(g, h, H := h^\beta, e(g,h)^\alpha)$ as the public key pk and (β, g^α) as the master secret key msk.[16]
- $\underline{\mathsf{KeyGen}(\mathsf{msk}, S)}$ Pick $r, r_y \leftarrow_R \mathbb{Z}_p$ for every $y \in S$. Then output

$$D := g^{(\alpha+r)/\beta},$$
$$\forall y \in S : \quad D_y := g^r \cdot \mathcal{H}(y)^{r_y}, \quad D'_y := h^{r_y}$$

as the key.
- $\underline{\mathsf{Encrypt}(\mathsf{pk}, (\mathbf{M}, \pi), \mathsf{msg})}$ Suppose $\mathbf{M}$ has n_1 rows and n_2 columns. Pick $s, v_2, \ldots, v_{n_2} \leftarrow_R \mathbb{Z}_p$ and let $\mathbf{v} = (s, v_2, \ldots, v_{n_2})$. Let $\mu_i = \langle (\mathbf{M})_i, \mathbf{v} \rangle$ for $i = 1, \ldots, n_1$, where $(\mathbf{M})_i$ denotes the ith row of $\mathbf{M}$. Then output

$$\tilde{C} := e(g,h)^{\alpha s} \cdot \mathsf{msg}, \quad C := H^s,$$
$$\forall i = 1, \ldots, n_1 : \quad C_i := h^{\mu_i}, \quad C'_i = \mathcal{H}(\pi(i))^{\mu_i}$$

as the ciphertext.
- $\underline{\mathsf{Decrypt}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk})}$ Let $\{\gamma_i\}_{i \in I}$ be a set of constants that exist when the set of attributes S in sk satisfies the MSP $(\mathbf{M}, \pi)$ in ct. Compute

$$\mathsf{prod} := \prod_{i \in I} \left[\frac{e(D_{\pi(i)}, C_i)}{e(C'_i, D'_{\pi(i)})} \right]^{\gamma_i}$$

and output $(\tilde{C} \cdot \mathsf{prod})/e(D, C)$.

E WATERS CP-ABE SCHEME

Below is the version of Waters' CP-ABE scheme [57, Section 3] that we implemented in asymmetric groups. Let $\mathcal{U} = \{1, 2, \ldots, U\}$ be the universe of attributes.

- $\underline{\mathsf{Setup}(1^\lambda)}$ Run $\mathsf{GroupGen}(1^\lambda)$ to obtain $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h)$. Pick $\alpha, a \leftarrow_R \mathbb{Z}_p$ and $G_1, \ldots, G_U \leftarrow_R \mathbb{G}$. Output $(g, h, g^a, e(g,h)^\alpha, G_1, \ldots, G_U)$ as the public key pk and g^α as the master secret key msk.
- $\underline{\mathsf{KeyGen}(\mathsf{msk}, S)}$ Here S is a subset of $\mathcal{U}$. Pick $t \leftarrow_R \mathbb{Z}_p$. Then output

$$K := g^\alpha \cdot g^{at}, \quad L := h^t, \quad \forall y \in S : \quad K_y := G_y^t$$

as the key.
- $\underline{\mathsf{Encrypt}(\mathsf{pk}, (\mathbf{M}, \pi), \mathsf{msg})}$ Suppose $\mathbf{M}$ has n_1 rows and n_2 columns. Pick $s, v_2, \ldots, v_{n_2} \leftarrow_R \mathbb{Z}_p$ and let $\mathbf{v} = (s, v_2, \ldots, v_{n_2})$. Let $\mu_i = \langle (\mathbf{M})_i, \mathbf{v} \rangle$ for $i = 1, \ldots, n_1$. Also pick $r_1, \ldots, r_{n_1} \leftarrow_R \mathbb{Z}_p$. Then output

$$C := e(g,h)^{\alpha s} \cdot \mathsf{msg}, \quad C' := h^s,$$
$$\forall i = 1, \ldots, n_1 : \quad C_i := g^{a\mu_i} G_{\pi(i)}^{-r_i}, \quad D_i := h^{r_i}$$

as the ciphertext.
- $\underline{\mathsf{Decrypt}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk})}$ Let $\{\gamma_i\}_{i \in I}$ be a set of constants that exist when the set of attributes S in sk satisfies the MSP $(\mathbf{M}, \pi)$ in ct. Output

$$\frac{C \cdot e\left(\prod_{i \in I} C_i^{\gamma_i}, L\right) \cdot \prod_{i \in I} e(K_{\pi(i)}, D_i^{\gamma_i})}{e(K, C')}.$$

We have reorganized the terms slightly to improve decryption time: instead of 2 pairings and an exponentiation in the target group per attribute, we now have 1 pairing and one exponentiation each in the two source groups.

F GPSW KP-ABE SCHEME

Below is the version of GPSW's KP-ABE scheme [29, Appendix A.1] that we implemented in asymmetric groups. Let $\mathcal{U} = \{1, 2, \ldots, U\}$ be the universe of attributes.

- $\underline{\mathsf{Setup}(1^\lambda)}$ Run $\mathsf{GroupGen}(1^\lambda)$ to obtain $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h)$. Pick $\alpha, t_1, \ldots, t_U \in \mathbb{Z}_p$. Output $(T_1 := g^{t_1}, \ldots, T_U := g^{t_U}, Y := e(g,h)^\alpha)$ as the public key pk and $(h, t_1, \ldots, t_U, \alpha)$ as the master secret key msk.
- $\underline{\mathsf{KeyGen}(\mathsf{msk}, (\mathbf{M}, \pi))}$ Suppose $\mathbf{M}$ has n_1 rows and n_2 columns. Pick $v_2, \ldots, v_{n_2} \leftarrow_R \mathbb{Z}_p$ and let $\mathbf{v} = (\alpha, v_2, \ldots, v_{n_2})$. Let $\mu_i = \langle (\mathbf{M})_i, \mathbf{v} \rangle$ for $i = 1, \ldots, n_1$. Then output

$$\forall i = 1, \ldots, n_1 : \quad D_i := h^{\frac{\mu_i}{t_{\pi(i)}}}$$

as the key.
- $\underline{\mathsf{Encrypt}(\mathsf{pk}, S, \mathsf{msg})}$ Pick $s \leftarrow_R \mathbb{Z}_p$ and output

$$E' := Y^s \cdot \mathsf{msg}, \quad \forall y \in S : \quad E_y := T_y^s$$

as the ciphertext.
- $\underline{\mathsf{Decrypt}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk})}$ Let $\{\gamma_i\}_{i \in I}$ be a set of constants that exist when the set of attributes S in sk satisfies the MSP $(\mathbf{M}, \pi)$ in ct. Output

$$\frac{E'}{\prod_{i \in I} e(E_{\pi(i)}, D_i)^{\gamma_i}}.$$

[16]The group element f in their set-up algorithm is used only for delegation.