SPATIAL ENCRYPTION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Mike Hamburg

June 2011

Dedication

For the glory of God.

Acknowledgements

There are many people without whom this work would not have been possible. I would never have entered graduate school at Stanford, and having entered would never have finished, without the support of my family and friends. Beyond my family, I would especially like to thank the entering class of 2006 for their friendship throughout our five years here, and the Society for Freedom and Service to Humanity for being such a wonderful community.

For the material itself, I owe many thanks to Adam Barth, who proposed some of the central problems addressed here; to Dan Boneh, Xavier Boyen, Eujin Goh, Allison Lewko and Brent Waters, upon whose work the solutions are built; and to the rest of the security and theory groups, who have listened and contributed to many iterations of those solutions. Of course, Dan Boneh deserves the greatest acknowledgement in this area: not only did he teach me cryptography and guide my study of it, he also worked directly on spatial encryption.

Finally, the credit for this work belongs to the LORD Almighty. Beyond the "ordinary" miracles of creation and providence, His inspiration at key points in this work paved the path from a short, mediocre paper to a full thesis.

Contents

Chapter 1

Introduction

1.1 Public-Key Cryptography

The goal of a data encryption system is for two parties, traditionally named Alice and Bob, to communicate while assuring the secrecy and authenticity of their messages. Early cryptographic systems required Alice and Bob to share a secret key before encryption could take place. But this presented a problem: how are they to securely agree on a secret key, if they cannot communicate secretly without one? They would need another way to communicate, such as meeting in-person in a secure location.

Public-key encryption solves this problem. Each party generates a pair of keys: a *public key* and a *secret key*. A message encrypted using a public key can only be read using the corresponding secret key. Thus the keys live up to their names: the secret key must remain secret, but the public key can be published for all to see. Still, difficulties remain. Alice must acquire a copy of Bob's public key before she can send him a message, and she must confirm that it actually belongs to him, lest some malicious eavesdropper (Eve) substitute her own public key for Bob's.

1.2 Identity-Based Cryptography

Identity-based cryptography, proposed by Adi Shamir in 1984 [35], adds a twist to public-key cryptography: it allows any string to be used as a public key. For example,

if Alice wants to send an email to Bob, she might encrypt it using his email address, `bob@microsoft.com`, as the public key. She might just as well use any other piece of uniquely identifying information; anything that can be encoded as a string will work.

A little thought will show that this model is oversimplified. How is Bob to decrypt the message? For the encryption to be secure, he will need a secret key that nobody else knows, but he has had no input into the encryption process. The answer is that encryption also uses a master public key, called the *public parameters*. These parameters are generated by an authority called the *private key generator*, or PKG, together with a *master secret key*. After Bob proves his identity to the PKG, it uses the master secret key to give him a secret key corresponding to that identity.

An important side effect of this process is that unlike traditional cryptosystems, identity-based systems enforce *key escrow*. That is, the PKG can recreate Bob's secret key at any time, and so can read any messages sent to him – or to anybody else. In some contexts, key escrow is unacceptable. This is especially the case on the internet at large: why should some authority have the ability to read everyone's encrypted emails? But within an organization, key escrow may be desirable, both to recover lost key material and to respond to legal inquiries.

In any case, ordinary identity-based cryptography has an all-or-nothing escrow system: the single PKG has access to all secret keys, and each user has access to his own secret key, but there is nothing in between. We would prefer to allow a middle ground, where for example the PKG could *delegate* to Microsoft its power to make secret keys for strings ending in `@microsoft.com`. This design, called *hierarchical IBE* (HIBE), has the added convenience that Bob need not prove his identity to the PKG, but can instead prove it to Microsoft. This is considerably easier, since Microsoft created his account in the first place. Hierarchical IBE can also make escrow more acceptable. For example, a coalition of companies could set up an HIBE system, create delegated keys for themselves (covering only their own domains), and then destroy the master secret key. They would thus achieve the convenience of identity-based cryptography without centralized escrow.

In this paper, we will concern ourselves with such generalizations of identity-based cryptography. To distinguish the original version from them, we will call it *ordinary*

or *vanilla* IBE.

1.3 Related Work

Since the 2001 construction of IBE by Boneh and Franklin [8] and Cocks [16], many variants of IBE have been defined.

A hierarchical IBE, first defined by Horwitz and Lynn [22] and constructed by Gentry and Silverberg [20], allows some keys to be delegated to create other keys. Canetti, Halevi and Katz showed in [13] how to use an HIBE to construct forward-secure encryption, and in [14] how to use IBE or HIBE for chosen-ciphertext security. Our work is heavily based on the constant-size-ciphertext HIBE of Boneh, Boyen and Goh [7] and its subsequent extensions by Boneh and Hamburg [10] and by Lewko and Waters [26].

Further developments included more complex hierarchies and more complex rules for when a message could be decrypted. Yao, Fazio, Dodis and Lysyanskaya developed HIBE for complex hierarchies [38]. Additional flexibility comes from Sahai and Waters' fuzzy IBE [31], and the attribute-based encryption scheme of Goyal, Pandy, Sahai and Waters [21]. Bethencourt, Sahai and Waters constructed attribute-based encryption with policies attached to ciphertexts [5], and Ostrovsky, Sahai and Waters added non-monotonic formulae [29].

Predicate encryption is closely related to attribute-based encryption. Many schemes have been designed for inner product and similar predicates, including Okamoto and Takashima's hierarchical inner-product encryption [28]. This encryption system is related to our spatial encryption system, and the two are equivalent under a quadratically expensive reduction. Inner-product schemes can be improved to add support for boolean and polynomial equations, as shown by Katz, Sahai and Waters [23].

An anonymous IBE conceals not only the message but also the identity of the recipient. Boneh and Franklin's IBE is anonymous under suitable assumptions, and several authors developed further anonymous IBE schemes, including Boyen [11]; Boneh, Gentry and Hamburg [9]; and Ducas [18]. Boyen and Waters also developed an anonymous HIBE [12]. Shen, Shi and Waters developed an anonymous predicate

encryption system [36]. Beyond traditional anonymity, Caro, Iovino and Persiano show how to keep recipients' identities secret from other recipients [17].

Proof techniques for IBE and similar systems have improved over the last few years. Waters' dual-system encryption [37] enables proofs of adaptive security. This has been adapted by Lewko and Waters [26] and by Lewko, Okamoto, Sahai, Takashima and Waters [25] to prove adaptive security of several of the above designs. Attrapadung and Libert used dual-system technology to extend and further secure spatial encryption [3].

Recent work has also built several of these primitives from lattices, beginning with Gentry, Peikert and Vaikuntanathan [19]. It is harder to delegate lattice-based secret keys than it is to delegate group-based ones, but recent work is beginning to overcome this problem. This includes work by Agrawal, Boneh and Boyen on HIBE and related delegation problems [1, 2], and by Cash, Hofheinz, Kiltz and Peikert, also on delegation [15].

1.4 Motivation

The original motivating problem for our work was proposed by Adam Barth. Email encryption is commonly cited as a useful application of IBE, and yet almost nobody uses IBE to encrypt email. A major reason for this is the need to trust the PKG. This problem would be mitigated, though not completely removed, if users were given a choice of PKGs to trust. This is analogous to public-key infrastructures such as X.509, where users can choose which certificate authorities they trust.

Suppose that Alice trusts n different PKGs, and wishes to encrypt a message to u other users. That is, she wants for each of the u users to be able to decrypt the message if he has a key, attesting his identity, which was generated by one of the n trusted PKGs. She could encrypt once for each user, for each PKG, but this would take $n \cdot u$ encryptions. Using identity-based multicast encryption [32], she could cut this down to n encryptions, but would still need to do $O(n \cdot u)$ work. We would like to do better, ideally doing $O(n + u)$ work and creating a ciphertext of constant size.

We can go further by adding more desiderata. We might hope for a HIBE-like

design, where Microsoft can generate a key for `bob@microsoft.com`. Here Microsoft's key would have been generated by some PKG P, and Bob's key would carry the approval of the same P, as if P had generated the key itself. This implies a *hierarchical multicast IBE*; no such scheme was known prior to this work.

We might also want keys to expire after some time, or conversely, for newer keys to be unable to decrypt older messages. Then if Bob's key is compromised, the attacker will not be able to decrypt much of his email with it. Such *forward-secure* constructions are known from hierarchical IBE [13], but no forward-secure multicast scheme was known prior to this work.[1]

To address these desiderata, we will need to design a modular identity-based encryption system, one which can be adapted to many different purposes. Such schemes are already known, and include multiple-HIBE [38] and attribute-based encryption [21]. We hope for our system to be more flexible than these prior systems, and additionally to produce shorter ciphertexts.

1.5 Comparison to Public-Key Cryptosystems

It is well-known that most users, whether individuals or corporations, do not readily adopt cutting-edge cryptographic technologies. Instead, they tend to build solutions from a handful of well-established primitives: symmetric and public-key encryption, message authentication codes, signatures and hash functions. This has several advantages: software for these primitives is more readily available on many platforms, many programmers have experience with them, and compliance criteria may specify the use of specific ciphers. Therefore, it is important to consider the trade-offs between identity-based cryptosystems and traditional ones.

Nearly any identity-based cryptosystem can be emulated using a "rights management" server. This server holds the secret key for a public-key cryptosystem which

[1] We make a distinction between multicast and broadcast schemes, using the former term for cryptosystems in which users are explicitly included, and the latter for those in which they are explicitly excluded. Forward secure broadcast encryption was constructed in [38]. Note that [32] is, in our terminology, a multicast design rather than a broadcast one.

resists chosen-ciphertext attacks (that is, messages encrypted with that system cannot be modified, only replaced entirely). To encrypt a message under some policy, a sender simply concatenates the policy and the message, then encrypts both using the server's public key. Alternatively, the policy may be sent in the clear, with only its hash present in the message. To decrypt, a recipient presents the server with the encrypted message and with some credential which demonstrates his right to read the message. The server decrypts the message, checks that the credential suffices under the policy, and returns the decrypted message. By using a hybrid encryption mode, in which a symmetric key is encrypted with a public-key cryptosystem, the server's load is kept reasonable and users retain some measure of privacy, as the server need only decrypt that key.

Such a system has significant advantages over an identity-based encryption system. The server can audit accesses, either to uncover employees who abuse their power or to assess damage in the case of a compromise. It can also revoke access, which is impossible in an identity-based cryptosystem. Perhaps more appealing is a combination of the two: a tripwire system separate from the rights management server can read the audit logs in real time; access to large amounts of sensitive data could set off alarms, and the server could temporarily or permanently block access from the suspect user. Furthermore, such a system meets the highest standards of security for which identity-based systems strive: users cannot gain by collusion, policies may be kept secret even from recipients, revocation is absolute, and an attacker cannot hope to find weak policies.

However, there are also significant disadvantages to the server-based approach. The simplest is that for some users, auditing and revocation may be liabilities rather than benefits, and they are difficult to remove from a rights-management system. But even when these are desirable features, the rights management server is a single point of failure both for service and for security. It must be online at all times, but at the same time must resist all attacks. By contrast, the PKG in an identity-based cryptosystem need not be online continuously (or at all), and it handles a much lower volume of traffic, so its attack surface is much less. Thus, even when a rights-management system is desirable, it can be shored up by additionally encrypting with

an identity-based system.

1.6 Efficiency

Much of our work concerns algorithms which we hope to run on an actual computer. As such, we require those algorithms to be *efficient* according to some measure of efficiency. Formally, we require that each algorithm takes a security parameter, traditionally denoted by λ, and that all our algorithms take time which is polynomial in λ. Furthermore, many of our algorithms are parameterized by a *system complexity parameter* called n. This parameter is generally related to the maximum size of a set, the maximum depth of a hierarchy, etc. We require algorithms which take an n to run in time polynomial in both n and λ. Similarly, we require all the objects in the system to be efficiently encoded in a computer memory, so that in particular they take $\text{poly}(n, \lambda)$ bits to represent.

In practice, polynomial time is not very specific. Users set λ as some number of "estimated bits of security", usually between 80 and 256. For any such λ and for reasonable n (say, $n < 1000$), we want the algorithms that are part of our cryptographic schemes to be fast, taking at most a few seconds to run on a modern computer but preferably much less. We would like to achieve modest memory usage, perhaps tens of kilobytes for secret keys, less than a kilobyte for ciphertexts, and a few megabytes for computations. On the other hand, we allow the attacker a significantly longer time, approximately on the order of 2^λ, corresponding to years or longer on a large supercomputer. We do not call such a large computation efficient, but rather *feasible* – though once again, formally speaking a feasible computation is one which takes time polynomial in λ and n. Similarly, we can define a *reasonable* number of queries or a *negligible* probability.

This intuitive definition suffices because we prove security concretely. That is, we show that an algorithm which breaks our cryptosystem can be adapted to solve some (hopefully) difficult mathematical problem – though perhaps with a somewhat lower probability – by performing only a few extra steps.

1.7 Notation and Mathematical Background

We introduce most of our notation and mathematical background here, but leave a few technical points to the sections in which they are used.

By $\mathbb{Z}_p$ we mean the field of integers modulo a prime number p. We sincerely apologize to any mathematically inclined readers who feel that this symbol should be reserved for the p-adic integers. The symbol $\mathbb{Z}_p^*$ denotes the nonzero integers modulo p.

We use calligraphic symbols ($\mathcal{G}, \mathcal{S}, \mathcal{T}$) to identify sets and groups, and also to define adversaries ($\mathcal{A}$, $\mathcal{B}$). The set of integers between a and b inclusive we denote by $[a, b]$; to exclude b we write $[a, b)$. We use capital fraktur symbols ($\mathfrak{A}, \mathfrak{B}, \mathfrak{C}$) for cryptographic hierarchies (whose meaning we will describe shortly) and lower-case fraktur symbols ($\mathfrak{a}, \mathfrak{b}, \mathfrak{c}$) for their elements.

We use the capital letters U, V and W for affine subspaces of a vector space, and K, L and M for matrices. In a block matrix, written $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$, a blank entry indicates a zero. We use overarrows ($\vec{v}$) to denote vectors. Vectors in this paper are always column vectors, but to save space we write them as transposed row vectors, for example: $\langle 1, 2, 3 \rangle^\top$. By $\vec{u}^\top \vec{v}$ we mean the inner product of the vectors $\vec{u}$ and $\vec{v}$.

Essentially all our groups are abelian. We will write these groups in additive notation (meaning that the group operation is $+$ and the identity element is 0). We use Greek letters to identify group elements, except for λ which is a security parameter, χ which is a setup parameter, and ϵ which is a vanishingly small quantity.

By $\{0, 1\}^*$, we mean the set of all (binary-encoded) strings of arbitrary length. We use H to denote cryptographic (collision-resistant) hash functions. By $\Pr(e)$ we mean the probability that a given event e occurs.

We write algorithm names in SMALL CAPS, and identifiers such as domain names in `teletype`. We write data scrutures in ALL CAPS, and implementations of cryptographic systems in **bold**.

1.8 Random Oracles and Generic Groups

Cryptographic proofs strive to consider all possible attacks against a given system. But in practice, most attacks share certain characteristics. Therefore, in places where a complete security proof is too difficult, we will show security against attacks with particular, common characteristics. This is especially useful for base assumptions. No practical cryptosystem supports an unconditional proof of security, so we will need to assume that certain problems are difficult in order to proceed. To substantiate our belief that these problems are difficult, we will show that *generic* techniques cannot solve them.

The two models we will use in this paper are the *random oracle model* and the *generic group model*.

Random oracle model The random oracle model assumes that the adversary treats a particular function as a black box, i.e. as a random function from its domain to its codomain. Most cryptographic hash functions are designed to behave as similarly as possible to random oracles, so we will occasionally treat them as such. This is an imperfect model of reality, however, because any real hash function has a compact description and is efficiently computable.

The main advantage of the random oracle model is that if an attacker treats the hash function as a random oracle, her attack will still work if it is replaced by a different function that produces the same distribution. For example, if the hash function H returns an element of a cyclic group $\mathcal{G}$ (generated by α), it could be replaced by a function that chooses a random $x \in [0, |\mathcal{G}|)$ and returns $x \cdot \alpha$. An attack against a system modified in this way might more easily reduce to a hard mathematical problem than one which uses H.

A second use is to force the adversary's choices. Suppose that we are interested in the case when one of the adversary's choices (say, a message to sign) is a certain random number r. If the message is passed through a random oracle, we can rig that oracle to return r for one of the adversary's queries. Then if the adversary makes q queries to the random oracle, she will choose the favored r with probability $1/q$.

A third use is to shorten randomly-generated parameters. Some systems' setup phases require generating and publishing several random parameters; it is more compact to declare them to be the outputs $H(1), H(2), \ldots, H(n)$ for some hash H. If H is modeled as a random oracle, then these procedures are effectively the same.

Most results in this work do not use the random oracle model. We use it only for optional extensions and to strengthen certain security proofs for signature schemes.

Generic group model Much of the arithmetic in this paper is done over groups of points on an elliptic curve. In practice, most known attacks on such systems do not use the elliptic curve structure; they treat the group operation and the representation of the elements as black boxes. Given multiple groups supporting a bilinear map, the same is usually still true so long as the map's output is large enough to avoid attacks by index calculus. Therefore, it makes sense to consider what generic attacks can accomplish. Because this model robs the attacker of much of her power, we will only use it to defend our base assumptions.

Chapter 2

Generalized Identity-Based Encryption

2.1 Examples

In this chapter, we will propose a model for variants of IBE. We begin by giving examples of several such variants which have appeared in the literature.

Multicast IBE In multicast IBE, as in ordinary IBE, each user has an identity and a corresponding secret key. However, when encrypting a message, the sender can choose several identities to encrypt to, and a secret key corresponding to any of those identities suffices to read the message. This is easily accomplished by encrypting the message once for each user, but multicast IBE is more efficient in computation, space usage or both.

This form of IBE is often called *broadcast IBE*. However, we wish to distinguish it from another definition of broadcast encryption, in which the sender wishes to send a message to every user except for those on a short list. Such a model is useful for revoking compromised devices in a digital content distribution system. We will also model broadcast IBE.

Hierarchical IBE In HIBE, identities are arranged in a hierarchy, encoded as sequences of strings. If s is an intial subsequence of a sequence t, then a key for s can be delegated to create a key for t, and so can also be used to decrypt messages which were encrypted to t. For example, a key for $(\texttt{com}, \texttt{microsoft})$ can be delegated to a key for $(\texttt{com}, \texttt{microsoft}, @, \texttt{bob})$, which we might use as an encoding for Bob's email address `bob@microsoft.com`.

Predicate Encryption Secret keys in predicate encryption correspond to strings. When encrypting a message, a sender chooses a function $f : \{0,1\}^* \to \{0,1\}$, possibly from restricted set of such functions. The message can be decrypted with the key for a string s where $f(s) = 1$. Alternatively, the string may be chosen at encryption time, and predicate may be associated with the key.

Anonymous IBE In most IBE systems, anyone can determine (or at least verify) the identities of the intended recipients simply by looking at the ciphertext. An anonymous IBE system prevents this. For more advanced systems such as predicate encryption, in which messages are not encrypted using individual identities, such a system is often described as *key-private*, *policy-private* or something similar.

2.2 Overview of the Model

To study such systems, we first provide a model of what they accomplish, and of the security guarantees we desire. We call this the Generalized Identity-Based Encryption (GIBE) model.

Roles Users in a GIBE can take on various *roles*, chosen from some set $\mathfrak{R}$ of allowable roles. We place no restrictions on $\mathfrak{R}$. For each role $\mathfrak{r}$, we assign a distribution $\mathcal{SK}_{\mathfrak{r}}$ of secret keys corresponding to that role. For example, in ordinary IBE and multicast IBE, users' roles are their identities (which are simply strings). Additionally, the PKG has a role, which we denote $\top$, and holds the corresponding master secret key $\mathrm{SK}_{\top}$.

Delegation Keys for certain roles can be delegated to create keys for other roles. We say that a role $\mathfrak{r}_1$ *delegates* to a role $\mathfrak{r}_2$, and write $\mathfrak{r}_1 \succeq \mathfrak{r}_2$, if a key for $\mathfrak{r}_1$ can be used (by an efficient algorithm DELEGATE) to create a key for $\mathfrak{r}_2$. This relation is transitive, meaning that if $\mathfrak{r}_1 \succeq \mathfrak{r}_2$ and $\mathfrak{r}_2 \succeq \mathfrak{r}_3$ then $\mathfrak{r}_1 \succeq \mathfrak{r}_3$.[1] What is more, if $\mathfrak{r}_1 \succeq \mathfrak{r}_2 \succeq \mathfrak{r}_1$, then $\mathfrak{r}_1$ and $\mathfrak{r}_2$ are essentially equivalent: a key for either one is as good as a key for the other. Therefore we call such roles equal, meaning that $\succeq$ is antisymmetric. Thus $\succeq$ defines a *partial order* on $\mathfrak{R}$ (which makes $\mathfrak{R}$ a *partially-ordered set*, or "poset").

Policies Let $\mathfrak{P}$ be a set of allowable policies. We place no restrictions on $\mathfrak{P}$. When a user encrypts a message using a GIBE, she chooses a *policy* $\mathfrak{p} \in \mathfrak{P}$. The chosen policy governs which users will be able to decrypt the message using the keys corresponding to their roles. We say that $\mathfrak{r}$ *decrypts* a policy $\mathfrak{p}$, and write $\mathfrak{r} \succeq \mathfrak{p}$, if a key $\mathrm{SK}_{\mathfrak{r}}$ for $\mathfrak{r}$ can be used to decrypt a message encrypted using $\mathfrak{p}$ (using an efficient algorithm DECRYPT). This relation is transitive with delegation, so that (if we also say that $\mathfrak{p} \succeq \mathfrak{p}$) the sum $\mathfrak{R} \coprod \mathfrak{P}$ is also a poset.

Notice that the decryption relation checks if a single role decrypts a single policy. Our model implicitly requires collusion resistance: for a user holding the secret keys to multiple roles $\mathfrak{r}_1, \ldots, \mathfrak{r}_m$ to decrypt a message under $\mathfrak{p}$, we must have $\mathfrak{r}_i \succeq \mathfrak{p}$ for some $i \in [1, m]$. In particular, we do not model threshold encryption schemes, in which multiple keys are required to decrypt a message.

Together, we call the sets $(\mathfrak{R}, \mathfrak{P})$ and relation $\succeq$ a *hierarchy*, which we denote $\mathfrak{H}$.

Simplifications To simplify analysis, we will place a few additional requirements on the model. We will require that the systems have a role $\top$, corresponding to the master PKG, such that $\top \succeq \mathfrak{r}$ for all roles $\mathfrak{r} \in \mathfrak{R}$ and $\top \succeq \mathfrak{p}$ for all policies $\mathfrak{p} \in \mathfrak{P}$. In practice, this requirement is generally vacuous. Most systems have such a role already, and if they don't, it can be defined even though nobody holds the key in real life. A key $\mathrm{SK}_{\top}$ can generally be defined, at worst, by all the randomness used in setting up the system.

[1] One can imagine a system in which $\mathfrak{r}_1 \not\succeq \mathfrak{r}_3$ because the requisite $\mathfrak{r}_2$ is hard to find. We do not model such systems.

We assume that roles and policies can be efficiently encoded, and that the $\succeq$ relation can be determined efficiently.

More onerously, we require that when DELEGATE creates a key $\mathrm{SK}_{\mathfrak{r}}$ for a role $\mathfrak{r}$, it must generate $\mathrm{SK}_{\mathfrak{r}}$ uniformly from $\mathcal{SK}_{\mathfrak{r}}$. In other words, all keys for a given role come from the same distribution, no matter how they were delegated. Similarly, we require DECRYPT for some policy $\mathfrak{p}$ to return the same distribution of results for a given ciphertext, no matter what role $\mathfrak{r} \succeq \mathfrak{p}$ and corresponding key $\mathrm{SK}_{\mathfrak{r}} \in \mathcal{SK}_{\mathfrak{r}}$ are used to decrypt it. These requirements make the security model considerably simpler. In particular, this means that secret keys can be re-randomized, because every role delegates to itself. In this special case the requirement is unnecessary, but the designs discussed in this paper still satisfy it.

Finally, we will assume that all messages are fixed-length rather than arbitrary-length, so that length cannot be used to distinguish between two ciphertexts. This accurately reflects practice: asymmetric cryptosystems are used almost exclusively to encrypt symmetric (e.g. AES) secret keys, so they act more as *key encapsulation mechanims* (KEMs) than as encryption systems. Similarly, signature schemes are used almost exclusively to sign the hashes of messages.

Parameters The hierarchy $\mathfrak{H} = (\mathfrak{R}, \mathfrak{P})$ is not necessarily fixed in a given cryptosystem. For example, there may be a maximum multicast set size, maximum hierarchy depth, or some similar complexity parameter, and the roles and policies may depend on the security parameter. Such parameters we call χ_{user}. Similarly, there may be purely mechanical constraints on roles and policies. For example, roles in inner-product encryption may incorporate numbers modulo a prime p which is generated at system setup time. We call such parameters χ_{system}. To accomodate these variations, $\mathfrak{H}$ is indexed by $\chi := (\lambda, \chi_{\text{user}}, \chi_{\text{system}})$. For brevity, we will omit χ when possible.

2.3 Algorithms

A GIBE's implementation consists of four efficient randomized algorithms.

Setup To initialize the public parameters PP and the master secret key $\mathrm{SK}_\top$, the system's creators run an algorithm

$$(\mathrm{PP}, \mathrm{SK}_\top) \leftarrow \textsc{Setup}(\lambda, \chi_{\mathrm{user}})$$

For simplicity, we will assume that the system parameters $\chi = (\lambda, \chi_{\mathrm{user}}, \chi_{\mathrm{system}})$ are included as part of PP.

Delegate To create a secret key $\mathrm{SK}_{\mathfrak{r}_1}$ for a role $\mathfrak{r}_1$, there is an algorithm

$$\mathrm{SK}_{\mathfrak{r}_1} \leftarrow \textsc{Delegate}(\mathrm{PP}, \mathfrak{r}_2, \mathrm{SK}_{\mathfrak{r}_2}, \mathfrak{r}_1)$$

This algorithm requires that $\mathfrak{r}_2 \succeq \mathfrak{r}_1$, and must choose $\mathrm{SK}_{\mathfrak{r}_1}$ uniformly from $\mathcal{SK}_{\mathfrak{r}_1}$. Note that $\mathrm{SK}_{\mathfrak{r}_2}$ and $\mathfrak{r}_2$ are passed in separately. This is done so that we may talk about the size of secret key $\mathrm{SK}_{\mathfrak{r}}$ independently of the size of the description of $\mathfrak{r}$.

Encrypt To encrypt a message m under a policy $\mathfrak{p}$, a sender runs

$$C \leftarrow \textsc{Encrypt}(\mathrm{PP}, \mathfrak{p}, m)$$

Decrypt To decrypt a ciphertext C which was encrypted under a policy $\mathfrak{p}$, there is an algorithm

$$m \leftarrow \textsc{Decrypt}(\mathrm{PP}, \mathfrak{r}, \mathrm{SK}_{\mathfrak{r}}, \mathfrak{p}, C)$$

As in Delegate, $\mathfrak{r}$ and $\mathfrak{p}$ are passed separately from the ciphertext, so that we can talk about the size of the ciphertext independently from the size of its policy. Decrypt can fail by returing a special symbol $\perp$. For correctness, we require that if

$$\begin{aligned} C &\leftarrow \textsc{Encrypt}(\mathrm{PP}, \mathfrak{p}, m) \\ m' &\leftarrow \textsc{Decrypt}(\mathrm{PP}, \mathfrak{r}, \mathrm{SK}_{\mathfrak{r}}, \mathfrak{p}, C) \end{aligned}$$

where $\mathrm{SK}_{\mathfrak{r}} \in \mathcal{SK}_{\mathfrak{r}}$ and $\mathfrak{r} \succeq \mathfrak{p}$, then $m' = m$ (and in particular that decryption does not fail). Ordinarily require total correctness, but in some cases a negligible probability

of failure may be acceptable.

Although DECRYPT requires that $\mathfrak{r} \succeq \mathfrak{p}$, its output must be otherwise independent of $\mathfrak{r}$ and of $\mathrm{SK}_{\mathfrak{r}}$. If there is a minimum role $\mathfrak{r}' \succeq \mathfrak{p}$ (meaning that for all $\mathfrak{r} \succeq \mathfrak{p}$ we also have $\mathfrak{r} \succeq \mathfrak{r}'$), then by running

$$\begin{aligned} \mathrm{SK}_{\mathfrak{r}'} &\leftarrow \textsc{Delegate}(\mathrm{PP}, \mathfrak{r}, \mathrm{SK}_{\mathfrak{r}}, \mathfrak{r}') \\ C &\leftarrow \textsc{Decrypt}(\mathrm{PP}, \mathfrak{r}', \mathrm{SK}_{\mathfrak{r}'}, \mathfrak{p}, C) \end{aligned}$$

we can achieve this property generically: $\mathfrak{r}'$ does not depend on $\mathfrak{r}$, and $\mathrm{SK}_{\mathfrak{r}'}$ is uniformly random in $\mathcal{SK}_{\mathfrak{r}'}$ regardless of $\mathfrak{r}$ and $\mathrm{SK}_{\mathfrak{r}}$.

2.4 Security Games

We follow the tradition of defining the security of a GIBE in terms of a game between two players, a *challenger* and an *adversary* $\mathcal{A}$. The challenger represents the functionality that the system provides, while the adversary is attempting to go beyond that functionality by reading (or learning information about) a message that she[2] is not supposed to have access to. To show that she can learn information from ciphertexts, the adversary attempts to pass an *indistinguishability* or *semantic security* test: to determine from a ciphertext which of two chosen messages has been encrypted.

To support the several security models offered by IBE variants, we propose a security game which itself has several variants. We begin with the strongest variant. The game begins with a fixed λ and χ_{user}. It proceeds in several stages:

Setup The challenger runs SETUP to generate public parameters PP and a master secret key $\mathrm{SK}_{\top}$. It sends PP to the adversary.

First Query Phase The adversary can now make any (polynomial) number of queries to the challenger. Two types of queries are allowed:

[2]Adversaries get a feminine pronoun because they follow in the footsteps of the eavesdropper Eve. The challenger is more robotic, and so is given an undignified "it".

- Given a *delegation query* ("Delegate", $\mathfrak{r}$) where $\mathfrak{r} \in \mathfrak{R}$, the challenger must run

$$\mathrm{SK}_{\mathfrak{r}} \leftarrow \textsc{Delegate}(\mathrm{PP}, \top, \mathrm{SK}_{\top}, \mathfrak{r})$$

  and return $\mathrm{SK}_{\mathfrak{r}}$ to the adversary. Because of our restriction on DELEGATE, the returned $\mathrm{SK}_{\mathfrak{r}}$ is simply a uniform element of $\mathcal{SK}_{\mathfrak{r}}$.

- Given a *decryption query* ("Decrypt", $\mathfrak{p}, C$) where $\mathfrak{p} \in \mathfrak{P}$, the challenger must run

$$m \leftarrow \textsc{Decrypt}(\mathrm{PP}, \top, \mathrm{SK}_{\top}, \mathfrak{p}, C)$$

  and return m to the adversary. Because of our restriction on DECRYPT, this decryption might as well have been from any $\mathfrak{r} \succeq \mathfrak{p}$ and any secret key for that role.

Challenge At some point, the adversary chooses two policies $\mathfrak{p}_0$ and $\mathfrak{p}_1$ and two corresponding messages m_0 and m_1, and sends them to the challenger. She must not have asked a delegation query to any role $\mathfrak{r}$ such that $\mathfrak{r} \succeq \mathfrak{p}_0$ or $\mathfrak{r} \succeq \mathfrak{p}_1$. The challenger chooses $b \xleftarrow{\mathrm{R}} \{0, 1\}$, runs

$$C_* \leftarrow \textsc{Encrypt}(\mathrm{PP}, \mathfrak{p}_b, m_b)$$

and returns the challenge ciphertext C_* to the adversary. We hope that the encryption so completely hides the features of the message m_b that the adversary cannot guess which b was chosen with probability significantly greater than $\frac{1}{2}$.

Second Query Phase The second query phase is nearly the same as the first. To prevent trivial "attacks", the adversary cannot ask a delegation query to any role $\mathfrak{r}$ where either $\mathfrak{r} \succeq \mathfrak{p}_0$ or $\mathfrak{r} \succeq \mathfrak{p}_1$, nor may she ask the challenger to decrypt C_* using any policy.

Guess The adversary outputs a guess $b' \in \{0, 1\}$. We say that the adversary wins the game if $b' = b$, and loses otherwise. We define the advantage $\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{G})$ of the

adversary $\mathcal{A}$ against the GIBE $\mathbf{G}$ by

$$\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{G}) = |\Pr(b' = b) - \Pr(b' \neq b)|$$

Advantage is defined this way so that it will be between 0 and 1, with 0 representing no break and 1 representing a complete break. Since we will use advantages extensively in this paper, we will also note that

$$\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{G}) = |\Pr(\mathcal{A} \text{ outputs 1 when } b = 0) - \Pr(\mathcal{A} \text{ outputs 1 when } b = 1)|$$

and that

$$\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{G}) = |2\Pr(\mathcal{A} \text{ wins}) - 1|$$

We would like to say that no feasible adversary $\mathcal{A}$ has a non-negligible advantage against $\mathbf{G}$, for suitable definitions of "feasible" and "negligible".

There are several variants of this game, corresponding to weaker notions of security.

Selective vs. Adaptive Security The adversary in this game can select her target *adaptively*. A real-life adversary might only be interested in a single target, or more likely a small list of targets. We can formalize this notion of *selective* security by requiring the adversary to choose $\mathfrak{p}_0$ and $\mathfrak{p}_1$ before the game begins. To deal with a system parameter χ_{system} (upon which the policies depend), the challenger runs SETUP but only tells the adversary χ_{system}. The adversary chooses $\mathfrak{p}_0$ and $\mathfrak{p}_1$, then the challenger reveals the full PP and the game proceeds as normal.

We have a proof of adaptive security for one variant of our spatial encryption system. For another variant, we only have a proof of selective security.

Additionally, we can define a co-selective security game, in which the adversary chooses which roles she will query ahead of time. We will not use co-selective security in this paper, but it is used, for example, in [3].

Anonymity The above game enforces *anonymity*: not only is the adversary unable to determine any properties of the message m_b based on the ciphertext, she is also unable to determine any properties of the policy $\mathfrak{p}_b$. Most GIBEs in practice do not have this property. To model their non-anonymous security, we can add the constraint that $\mathfrak{p}_0 = \mathfrak{p}_1$.

We also note that a GIBE which is secure in the anonymous version of our game may still lack some aspects of anonymity. First, DECRYPT takes as input the policy $\mathfrak{p}$, which means that the recipient must know $\mathfrak{p}$. If DECRYPT uses $\mathfrak{p}$, we call the system *weakly anonymous*, and otherwise *strongly anonymous*.

Furthermore, our model does not cover systems which keep the recipients anonymous from each other. That is, the adversary is never allowed to be a recipient of the message, so our system makes no claims about what such an adversary can learn about the other recipients. We can model such a *recipient-anonymous* system with respect to a set R of roles by additionally allowing the adversary to make delegation queries for roles $\mathfrak{r} \in R$ where $\mathfrak{r} \succeq \mathfrak{p}_0$ and $\mathfrak{r} \succeq \mathfrak{p}_1$, so long as $m_0 = m_1$. The roles in R cannot be meaningfully delegated: if $\mathfrak{r} \succeq \mathfrak{r}'$, where $\mathfrak{r}' \succeq \mathfrak{p}_0$ but $\mathfrak{r}' \not\succeq \mathfrak{p}_1$, then testing decryption with $\mathfrak{r}'$ will break anonymity.

CPA-Only Security Our system allows decryption queries at any time, so it enforces full security against *chosen ciphertext attacks* (CCA_2). We can weaken this notion to CCA_1 security by disallowing decryption queries during the second query phase. CCA_1 models systems in which the ciphertext can be modified slightly without destroying its meaning. We can weaken this notion even further to chosen plaintext attacks (CPA) by disallowing decryption queries entirely.

We will show that for sufficiently powerful classes of GIBE, we can add CCA_2 security without making significant changes to the system.

2.5 Properties of GIBEs

GIBEs and their hierarchies have many useful properties and relations which we will use throughout the paper. This section will explore several of them. Readers who are

familiar with category theory will notice that, using the definitions in this section, hierarchies form a *category*, meaning that there are maps between hierarchies, as well as products and sums (coproducts) which interact with maps in intuitive ways.

Embedding It is clear that some GIBEs can be used as building blocks to create other GIBEs. For example, HIBE and multicast IBE can clearly be used to implement vanilla IBE. This is done by encoding the roles and policies for one GIBE into the other. To encode a hierarchy $\mathfrak{H} = (\mathfrak{R}, \mathfrak{P})$ into another hierarchy $\mathfrak{H}' = (\mathfrak{R}', \mathfrak{P}')$, we need:

- A function $f : \mathfrak{R} \to \mathfrak{R}'$. This function must preserve the partial order, meaning that
$$f(\mathfrak{r}_1) \succeq f(\mathfrak{r}_2) \text{ if and only if } \mathfrak{r}_1 \succeq \mathfrak{r}_2$$
(f is thus a *poset embedding*)
- A function $g : \mathfrak{P} \to \mathfrak{P}'$. This function must preserve decryptability, meaning that
$$f(\mathfrak{r}) \succeq g(\mathfrak{p}) \text{ if and only if } \mathfrak{r} \succeq \mathfrak{p}$$

We call such a pair (f, g) a *hierarchy embedding*, and we say that it *embeds* $\mathfrak{H}$ into $\mathfrak{H}'$.

Now, if (f, g) embeds $\mathfrak{H}$ into $\mathfrak{H}'$, then given a GIBE $\mathbf{G}'$ implementing $\mathfrak{H}'$, we can construct a GIBE $\mathbf{G}$ implementing $\mathfrak{H}$. We simply apply f (resp. g) to all roles (resp. policies) in $\mathfrak{H}$ before calling any of DELEGATE, ENCRYPT or DECRYPT. Also, if $\mathfrak{H}$ and $\mathfrak{H}'$ are parameterized, we need efficient functions to translate χ_{user} to χ'_{user} before calling SETUP, and to translate χ'_{system} to χ_{system} afterwards. In this case, (f, g) will need to be an embedding from $\mathfrak{H}_\chi$ to $\mathfrak{H}'_{\chi'}$.

Lemma (Embedding Lemma)**.** *If* $\mathbf{G}$ *is implemented as above, then for any adversary* $\mathcal{A}$ *against* $\mathbf{G}$ *(in any variant of the security game), there is an adversary* $\mathcal{A}'$ *against* $\mathbf{G}'$*, running in about the same time as* $\mathcal{A}$*, such that*

$$\text{Adv}(\mathcal{A} \leftrightarrow \mathbf{G}) = \text{Adv}(\mathcal{A}' \leftrightarrow \mathbf{G}')$$

Proof. The algorithms of $\mathbf{G}$ are simply translated into calls to the algorithms of $\mathbf{G}'$. An algorithm $\mathcal{A}'$ which translates the queries of $\mathcal{A}$ in the same way constitutes an adversary against $\mathbf{G}'$. Since f and g preserve the partial order, the same restrictions apply to $\mathcal{A}$ and to $\mathcal{A}'$, and their win conditions are also the same, so their advantages are the same. □

For category theorists, embeddings are the morphisms in the category of hierarchies, and implementation is a contravariant functor.

Note that our definition is stronger than what is required for a secure implementation. For that, we can allow $e(\mathfrak{r}_1) \succeq e(\mathfrak{r}_2)$ even when $\mathfrak{r}_1 \not\succeq \mathfrak{r}_2$, so long as the same does not happen for policies. However, in most cases there will be a policy $\mathfrak{p}$ which can be decrypted by $\mathfrak{r}_2$ but not by $\mathfrak{r}_1$, so a secure implementation will require an embedding.

Hashed Embedding At a high level, many GIBEs are defined to work with strings, but their implementations hash the strings to fixed-size numbers. Therefore we would like an anolog of the Embedding Lemma where f and g make use of collision-resistant hash functions. In this case (f, g) is not an embedding, because the partial order will not be preserved if a hash collision occurs. Therefore some correctness and some security may be lost, but we hope that only a little will be lost.

Suppose that $\mathfrak{H}(S)$ is a hierarchy whose roles and policies contain elements of some set S (e.g. strings) in their descriptions, but these elements are used only in equality comparisons. (This makes $\mathfrak{H}$ a covariant functor for suitably chosen categories.) Let H be a hash function from S to some other set T. Then we can implement a GIBE $\mathbf{G}$ for $\mathfrak{H}(S)$ using an implementation $\mathbf{G}'$ of $\mathfrak{H}(T)$ by applying H to all the S-elements (i.e. by applying the functor).

If the function to decide $\succeq$ in $\mathfrak{H}(S)$ is monotone with respect to the string equality comparisons, then $\mathbf{G}$ will always be a correct implementation. If not, then $\mathbf{G}$ may occasionally fail, but only if a hash collision occurs. The security properties are analogous:

Lemma (Hashed Embedding Lemma)**.** *If* $\mathbf{G}$ *is implemented in this way, then for any adversary* $\mathcal{A}$ *against* $\mathbf{G}$ *(in any variant of the security game), there is an adversary*

$\mathcal{A}'$ against $\mathbf{G}'$ and a collision-resistance adversary $\mathcal{B}$ against H, both running in about the same time as $\mathcal{A}$, such that

$$\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{G}) \leq \mathrm{Adv}(\mathcal{A}' \leftrightarrow \mathbf{G}') + \mathrm{CR}\,\mathrm{Adv}(\mathcal{B} \leftrightarrow H)$$

Proof. The algorithm $\mathcal{B}$ runs $\mathcal{A}$, hashing all the strings used in its $\mathfrak{H}(S)$ policies. If a hash collision occurs, then $\mathcal{B}$ successfully finds a collision in H. Otherwise, the partial order in the part of $\mathfrak{H}(S)$ that was actually used is the same as that in $\mathfrak{H}(T)$. In this case, if $\mathcal{A}$ wins against $\mathbf{G}$ then the adversary $\mathcal{A}'$ from the Embedding Lemma will also win against $\mathbf{G}'$. □

Products of Hierarchies Hierarchies support a natural notion of *products*. For two hierarchies $\mathfrak{H}_1 = (\mathfrak{R}_1, \mathfrak{P}_1)$ and $\mathfrak{H}_2 = (\mathfrak{R}_2, \mathfrak{P}_2)$, define their product $\mathfrak{H}_1 \times \mathfrak{H}_2$ as follows:

- Its roles are elements of $\mathfrak{R}_1 \times \mathfrak{R}_2$, i.e. pairs $(\mathfrak{r}_1, \mathfrak{r}_2)$ where $\mathfrak{r}_1 \in \mathfrak{R}_1$ and $\mathfrak{r}_2 \in \mathfrak{R}_2$. Their order is according to the product poset:

$$(\mathfrak{r}_1, \mathfrak{r}_2) \succeq (\mathfrak{r}'_1, \mathfrak{r}'_2) \text{ if and only if } \mathfrak{r}_1 \succeq \mathfrak{r}'_1 \text{ and } \mathfrak{r}_2 \succeq \mathfrak{r}'_2$$

- Likewise, its policies are elements of $\mathfrak{P}_1 \times \mathfrak{P}_2$, where

$$(\mathfrak{r}_1, \mathfrak{r}_2) \succeq (\mathfrak{p}_1, \mathfrak{p}_2) \text{ if and only if } \mathfrak{r}_1 \succeq \mathfrak{p}_1 \text{ and } \mathfrak{r}_2 \succeq \mathfrak{p}_2$$

By extension, we can define the product of any number of hierarchies. It is clear that products of hierarchies respect embeddings, so that this notion is natural.

A product of two hierarchies enforces both hierarchies simultaneously. It is comparable to double encryption, i.e. encrypting a message twice, using one policy from each hierarchy. However, double encryption is not collusion resistant. For example, suppose we wish to send a message to members of Lab 1 who have top secret clearances. We could encrypt the message under the policy "top secret", and then encrypt the resulting ciphertext to "members of Lab 1". But then two users, one with a top

secret clearance and one who is a member of Lab 1, could collude to decrypt it. In contrast, a GIBE implementing the product hierarchy would not allow this.

Coproducts of Hierarchies We can also define the *coproduct* of any number of hierarchies $\mathfrak{H}_1 = (\mathfrak{R}_1, \mathfrak{P}_1), \ldots, \mathfrak{H}_k = (\mathfrak{R}_k, \mathfrak{P}_k)$. The roles of the coproduct $\coprod_{i=1}^{k} \mathfrak{H}_i$ are either $\top$, or $(i, \mathfrak{r}_i)$ where $\mathfrak{r}_i \in \mathfrak{R}_i \backslash \{\top\}$. Its policies are $(i, \mathfrak{p}_i)$ where $\mathfrak{p}_i \in \mathfrak{P}_i$. Roles and policies are compared normally, but only when they have the same i.

While this notion of coproduct is natural, sometimes we want to give each hierarchy in the coproduct its own, separate $\top$. For example, the $\mathfrak{H}_i$ may represent divisions within a company, so that each division has its own hierarchy of power. We might then want to give each division head a master secret key, but only for that one division. For this case, we allow $\mathfrak{r}_i$ to be $\top$ in the above definition, and call the result an *extended coproduct*, which we write $\overline{\coprod}_{i=1}^{k} \mathfrak{H}_i$.

Extended coproducts (and thus ordinary coproducts as well) can be efficiently constructed using a product GIBE.

Lemma (Extended Coproduct Embedding)**.** *Let $\mathfrak{I}$ denote the hierarchy of vanilla IBE. Given hierarchies $\mathfrak{H}_1, \ldots, \mathfrak{H}_k$, all of which support embeddings e_i into another hierarchy $\mathfrak{H}_*$, there is an embedding from $\overline{\coprod}_{i=1}^{k} \mathfrak{H}_i$ into $\mathfrak{I} \times \mathfrak{H}_*$.*

This embedding is evident in the definition of coproducts: it simply maps $(i, \mathfrak{r}_i)$ to $(i, e_i(\mathfrak{r}_i))$, which is a valid member of $\mathfrak{I} \times \mathfrak{H}_*$, and likewise with the policies. The coproduct's $\top$ maps to $(\top, \top)$. Because this embedding is so efficient, it enables the main use of coproducts: to support many different schemes using the same set of public parameters.

There is a notable similarity between extended coproducts and HIBE schemes: in either one, there is a tag (or first path component) i, and several hierarchies underneath, one for each value of i.

2.6 Generic CCA_2 security

Product schemes give us a simple way to make schemes secure against chosen-ciphertext attacks. The approach is essentially the same as in [14]. We can use either a *one-time*

signature scheme, or a *weak commitment scheme* and a *one-time message authentication code*. The former approach is simpler, but the latter is more efficient, so we will show it here.

Weak Commitments A *weak commitment scheme* (or *encapsulation scheme* in the language of [14]) is a pair of algorithms: a randomized algorithm COMMIT which takes some object (say, a string) and outputs a pair (c, d) – a "commitment" and a "decommitment" – and a deterministic algorithm DECOMMIT which takes (c, d) and outputs either an object or $\perp$. For correctness, we require DECOMMIT $\circ$ COMMIT to be the identity.

There are two security properties. For the *binding* property, we require that if an honest committer generates a commitment to s, it should be hard to generate a decommitment to any $s' \neq s$. In other words, the adversary $\mathcal{A}$ plays the following game:

$$\begin{aligned} s &\leftarrow \text{random object} \\ (c, d) &\leftarrow \textsc{Commit}(s) \\ d' &\leftarrow \mathcal{A}(c, d) \end{aligned}$$

and wins if $\textsc{Decommit}(c, d') \notin \{\perp, s\}$; its advantage $\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathrm{bind})$ is its probability of winning.

For the *hiding* property, we require that it is hard to learn anything about s given c. The adversary $\mathcal{A}$ plays the following game:

$$\begin{aligned} s_0, s_1 &\leftarrow \text{random objects} \\ b &\xleftarrow{\mathrm{R}} \{0, 1\} \\ (c, d) &\leftarrow \textsc{Commit}(s_b) \\ b' &\leftarrow \mathcal{A}(s_0, s_1, c) \end{aligned}$$

and wins if $b' = b$; its advantage $\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathrm{bind})$ is $|\Pr(b = b') - \Pr(b \neq b')|$.

We call a weak commitment scheme *secure* if $\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathrm{bind})$ and $\mathrm{Adv}(\mathcal{B} \leftrightarrow \mathrm{hide})$

are negligible for all feasible adversaries $\mathcal{A}$ and $\mathcal{B}$. As usual, the definition of "feasible" and "negligible" may include an implicit security parameter λ.

One-Time MAC A one-time MAC scheme is an efficient deterministic function MAC which takes a random secret key s and an object x, and outputs a tag t. It should be difficult, given the tag for a chosen x, to compute the tag for any $x' \neq x$. Its security game is as follows:

- The adversary chooses an object x.
- The challenger chooses a random secret key s and sends $t = \mathrm{MAC}_s(x)$ to the challenger.
- The adversary computes an object x' and a tag t'.

The adversary wins if $t' = \mathrm{MAC}_s(x')$, and its advantage is its probability of winning.

Construction We now have the tools to construct a CCA_2 encryption scheme. Let $\mathfrak{H}$ be any hierarchy, and suppose we have a CPA-secure implementation $\mathbf{G}$ of $\mathfrak{H}$'s product $\mathfrak{H} \times \mathfrak{I}$ with the vanilla IBE $\mathfrak{I}$. To implement a GIBE for $\mathfrak{H}$ in a CCA_2-secure way, we will MAC all ciphertexts. Checking the MAC requires its secret key; this we place in the message before encrypting. To enforce the consistency of the system, we include a weak commitment to the MAC secret key in the encryption policy. The entire implementation is as follows:

- The SETUP algorithm is the same as in $\mathbf{G}$.
- The secret key for a role $\mathfrak{r} \in \mathfrak{H}$ is $\mathbf{G}$'s secret key for $(\mathfrak{r}, \top)$. To DELEGATE, simply append this $\top$ to the source and destination role, and pass through to $\mathbf{G}$'s DELEGATE algorithm.

- To implement $\textsc{Encrypt}(\mathrm{PP}, \mathfrak{p}, m)$, we use the following procedure:

$$\begin{aligned} s &\leftarrow \text{random MAC secret key} \\ (c,d) &\leftarrow \textsc{Commit}(s) \\ e &\leftarrow \textsc{Encrypt}_{\mathbf{G}}(\mathrm{PP},\ (\mathfrak{p},c),\ (d,m)) \\ a &\leftarrow \mathrm{MAC}_s(c,e) \\ &\quad \mathrm{return}(c,e,t) \end{aligned}$$

- The Decrypt algorithm reverses this procedure, checking the commitment and the MAC along the way. To implement $\textsc{Decrypt}(\mathrm{PP}, \mathfrak{r}, \mathrm{SK}_{\mathfrak{r}}, \mathfrak{p}, (c,e,t))$, we perform the following procedure:

$$\begin{aligned} (d,m) &\leftarrow \textsc{Decrypt}_{\mathbf{G}}(\mathrm{PP}, (\mathfrak{r},\top), \mathrm{SK}_{\mathfrak{r}}, (\mathfrak{p},c), e) \\ s &\leftarrow \textsc{Decommit}(c,d) \\ t' &\leftarrow \mathrm{MAC}_s(c,e) \\ &\quad \text{if } t' = t \text{ then return } m \text{ else return } \perp \end{aligned}$$

  Of course, if either $\textsc{Decrypt}_{\mathbf{G}}$ or Decommit returns $\perp$, then Decrypt also returns $\perp$.

If the GIBE and the commitment scheme are correct, then this algorithm is also clearly correct. The following theorem establishes its security.

Theorem 1 (Generic CCA_2 Security)**.** *Let $\mathbf{G}'$ be the GIBE defined as above. If $\mathbf{G}$ is CPA-secure, and the MAC and commitment schemes are secure, then $\mathbf{G}'$ is CCA_2-secure. If $\mathbf{G}$ enjoys anonymous security, then so does $\mathbf{G}'$, and if $\mathbf{G}$ enjoys adaptive security, then so does $\mathbf{G}'$.*

Proof. Our proof follows [14]. In addition to *winning*, we give the adversary two new objectives. One we call *bluffing*. Let the challenge ciphertext have commitment component c_*, a commitment to the MAC key s_*. We say that the adversary successfully bluffs if it makes a decryption query to (c_*, e, t), where either e or t is different from

the challenge ciphertext, but which decrypts to something other than $\perp$. (For consistency, the challenger can pick c_* ahead of time, so that bluffing has a meaning even before the adversary sees c_*.) We say that the adversary *forges* if it queries (c_*, e, t) where $t = \mathrm{MAC}_{s_*}(c_*, e)$, i.e. if the adversary successfully passes a MAC check on such a ciphertext. It is important to note that while detecting a bluff requires the ability to decrypt, detecting a forgery does not: it only requires knowledge of s_*.

Sequence of Games Our proof now proceeds by a *sequence of games*. In this proof technique, we will alter the security game, one piece at a time, until we reach a game which the adversary cannot win (or, in this case, bluff or forge). We will also argue that the adversary must perform the same way in each successive game, or else she has broken one of the primitives. Define w_i, b_i and f_i as the probability that the adversary wins, bluffs and forges, respectively, in Game i.

Game 0 Game 0 is the normal GIBE security game.

Game 1 Game 1 is the same as Game 0, except that the challenger rejects bluffs; that is, it returns $\perp$ for any decryption query whose $c = c_*$. This differs from Game 0 if and only if the adversary bluffs, so $|w_1 - w_0| \leq b_0$. But $b_1 = b_0$, because the games diverge only after $\mathcal{A}$ bluffs. Now, consider one of the bluffs. Either it is also a forgery, or else it produces a different MAC key $s \neq s_*$. In this second case, $\mathcal{A}$ has broken the commitment scheme. This creates an adversary $\mathcal{B}_1$ against the commitment scheme's binding property, where

$$\mathrm{Adv}(\mathcal{B}_1 \leftrightarrow \mathrm{bind}) \geq b_1 - f_1$$

There is another way to play Game 1, namely to emulate an attack on the underlying implementation $\mathbf{G}$. A simulator for the challenger in Game 1 can translate delegation queries on $\mathbf{G}'$ to those on $\mathbf{G}$. It can translate the challenge (creating the MAC key, the commitment and the MAC itself). The simulator can also translate decryption queries to (c, e, t) under a policy $\mathfrak{p}$, by creating a delegation query to $(\top, c)$, which can decrypt $(\mathfrak{p}, c)$ in the GIBE $\mathbf{G}$. Since in Game 1 decryption query is rejected if $c = c_*$, this role $(\top, c)$ is not above the policy $(\mathfrak{p}_*, c_*)$ used in the challenge, and so

is a legal query. This alternate way to play is identical as far as $\mathcal{A}$ is concerned.

Game 2 We define Game 2 to be the same as Game 1, except that $\textsc{Encrypt}_{\mathbf{G}}$ is called with the message $(0,0)$ instead of (d,m). Furthermore, for anonymous versions of the game, we encrypt both challenge messages to $\mathfrak{p}_0$. Since the two challenges are identical, $w_2 = \frac{1}{2}$. But if the adversary performs differently in Game 2 and Game 1, then it has broken $\mathbf{G}$. In particular, there are adversaries $\mathcal{B}_2$ and $\mathcal{B}_3$ such that

$$|w_1 - w_2| \leq \mathrm{Adv}(\mathcal{B}_2 \leftrightarrow \mathbf{G}) \quad \text{and} \quad |f_1 - f_2| \leq \mathrm{Adv}(\mathcal{B}_3 \leftrightarrow \mathbf{G})$$

Game 3 While $\mathcal{A}$ has no advantage in Game 2, we still need to bound her forgery abilities, to which end we define Game 3. In this game, the commitment c_* is replaced by a commitment to a random key (note that the decommitment d is no longer used). Then if $\mathcal{A}$ performs differently, it has broken the commitment scheme's hiding property. But if it does not, it has broken the MAC. This gives adversaries $\mathcal{B}_4$ and $\mathcal{B}_5$ against the commitment scheme and the MAC, where

$$f_2 \leq \mathrm{Adv}(\mathcal{B}_4 \leftrightarrow \mathrm{hide}) + \mathrm{Adv}(\mathcal{B}_5 \leftrightarrow \mathrm{MAC})$$

Summing up,

$$\begin{aligned} \frac{1}{2}\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{G}') &= \left| w_0 - \frac{1}{2} \right| \\ &\leq \mathrm{Adv}(\mathcal{B}_1 \leftrightarrow \mathrm{bind}) + \mathrm{Adv}(\mathcal{B}_2 \leftrightarrow \mathbf{G}) + \mathrm{Adv}(\mathcal{B}_3 \leftrightarrow \mathbf{G}) \\ &\quad + \mathrm{Adv}(\mathcal{B}_4 \leftrightarrow \mathrm{hide}) + \mathrm{Adv}(\mathcal{B}_5 \leftrightarrow \mathrm{MAC}) \end{aligned}$$

Because the various games respect our possible restrictions on the security game of $\mathbf{G}'$, the anonymity and adaptive security properties of $\mathbf{G}'$ are the same as those of $\mathbf{G}$. This completes the proof. □

Chapter 3

Signatures

The previous chapter concerned itself with generalized identity-based encryption. We can apply the same terminology to signatures, and use the same technology to build them. We call such schemes *generalized identity-based signature* (GIBS) schemes.

3.1 Algorithms

Like a GIBE, a GIBS operates on a hierarchy $\mathfrak{H} = (\mathfrak{R}, \mathfrak{P})$ and consists of four algorithms.

Setup and Delegate The SETUP and DELEGATE algorithms work the same as their GIBE counterparts.

Sign A user can sign a message m under a policy $\mathfrak{p}$ by calling

$$s \leftarrow \text{SIGN}(\text{PP}, \mathfrak{r}, \text{SK}_{\mathfrak{r}}, \mathfrak{p}, m)$$

where $\mathfrak{r} \succeq \mathfrak{p}$. Analogously to DECRYPT, we require the output of SIGN to be independent of $\mathfrak{r}$ and of $\text{SK}_{\mathfrak{r}}$.

Verify To verify the signature, any user can call

$$v \leftarrow \textsc{Verify}(\mathrm{PP}, \mathfrak{p}, m, s)$$

which returns 0 (for failure) or 1 (for success). For correctness, we demand that if

$$\begin{aligned} s &\leftarrow \textsc{Sign}(\mathrm{PP}, \mathfrak{r}, \mathrm{SK}_{\mathfrak{r}}, \mathfrak{p}, m) \\ v &\leftarrow \textsc{Verify}(\mathrm{PP}, \mathfrak{p}, m, s) \end{aligned}$$

where $\mathfrak{r} \succeq \mathfrak{p}$ and $\mathrm{SK}_{\mathfrak{r}} \in \mathcal{SK}_{\mathfrak{r}}$, then $v = 1$. As with decryption, a negligible loss of correctness may be acceptable.

3.2 Security Games

The security game for signatures is much simpler, containing only 3 phases.

Setup The challenger runs Setup to generate public parameters PP and a master secret key $\mathrm{SK}_{\top}$. It sends PP to the adversary.

Query Phase The adversary can now make any (polynomial) number of queries to the challenger. Two types of queries are allowed:

- Delegation queries work exactly the same as in a GIBE. Given a query ("Delegate", $\mathfrak{r}$) where $\mathfrak{r} \in \mathfrak{R}$, the challenger must run

  $$\mathrm{SK}_{\mathfrak{r}} \leftarrow \textsc{Delegate}(\mathrm{PP}, \top, \mathrm{SK}_{\top}, \mathfrak{r})$$

  and return $\mathrm{SK}_{\mathfrak{r}}$ to the adversary.

- Given a *signature query* ("Sign", $\mathfrak{p}, m$) where $\mathfrak{p} \in \mathfrak{P}$, the challenger must run

  $$s \leftarrow \textsc{Sign}(\mathrm{PP}, \top, \mathrm{SK}_{\top}, \mathfrak{p}, m)$$

and return s to the adversary. Because of our restriction on SIGN, it does not matter that the role used was $\top$; the same distribution of signatures would arise from any role and any secret key for that role.

Forgery The adversary outputs a policy $\mathfrak{p}$, a message m and a signature s. The adversary must not have queried ("Delegate", $\mathfrak{r}$) for any $\mathfrak{r} \succeq \mathfrak{p}$, nor ("Sign", $\mathfrak{p}, m$). The challenger then runs

$$v \leftarrow \textsc{Verify}(\mathrm{PP}, \mathfrak{p}, m)$$

The adversary wins if $v = 1$, and its advantage $\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{G})$ is defined as

$$\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{G}) = \Pr[v = 1]$$

As with GIBE, there are a number of weaker variants of the signature game.

Selective-Policy Security We can define a selective version of the game, in which the adversary must choose the policy $\mathfrak{p}$ to attack beforehand.

Selective-Message Security Similarly, we can require the adversary to choose the message beforehand. This corresponds to a more traditional notion of "selective forgery" (as compared to "existential forgery"). If the message is hashed before signing and the hash is modeled as a random oracle, then selective-message security is equivalent to adaptive-message security.

We could define other weaker notions, for example by preventing the adversary from making signature queries, but we see no profit in doing so.

Embedding and Hashed Embedding The Embedding Lemma and Hashed Embedding Lemma also apply to signature schemes, and it is clear that applying a collision-resistant hash to the message before signing is also secure.

3.3 Signatures from Encryption

It is often possible to build a signature scheme from an encryption scheme. Let $\mathfrak{H} = (\mathfrak{R}, \mathfrak{R})$ be a hierarchy whose roles are the same as its policies. Let $\mathbf{G}$ be an implementation of $\mathfrak{H} \times \mathfrak{I}$. Then we can use $\mathbf{G}$ to implement a signature scheme $\mathbf{S}$ on $\mathfrak{H}$. Secret keys for a role $\mathfrak{r}$ in the signature scheme will correspond to secret keys for $(\mathfrak{r}, \top)$, with delegation working in the natural way.

To implement $\textsc{Sign}(\mathrm{PP}, \mathfrak{r}, \mathrm{SK}_\mathfrak{r}, \mathfrak{p}, m)$, we set an identity i as an injective string encoding of the pair $(\mathfrak{p}, m)$, and run

$$s \leftarrow \textsc{Delegate}(\mathrm{PP}, (\mathfrak{r}, \top), \mathrm{SK}_\mathfrak{r}, (\mathfrak{p}, i))$$

Note that $\mathfrak{p}$ is also a role in $\mathfrak{H}$, because $\mathfrak{H}$'s policies are the same as its roles. The signature s is a secret key for $(\mathfrak{p}, i)$.

To implement $\textsc{Verify}(\mathrm{PP}, \mathfrak{r}, \mathrm{SK}_\mathfrak{r}, \mathfrak{p}, m, s)$, we compute i, choose a random message v and run:

$$\begin{aligned} c &\leftarrow \textsc{Encrypt}(\mathrm{PP}, (\mathfrak{p}, i), v) \\ v' &\leftarrow \textsc{Decrypt}(\mathrm{PP}, (\mathfrak{p}, i), s, (\mathfrak{p}, i), c) \\ &\text{if } v' = v \text{ then return 1; otherwise return 0} \end{aligned}$$

The correctness of this verification algorithm follows from the correctness of Decrypt. Likewise, the security of $\mathbf{S}$ follows from that of $\mathbf{G}$:

Theorem 2. *Suppose that* $\mathbf{G}$ *supports a large message space* $\mathcal{M}$*. If* $\mathbf{G}$ *is an adaptively CPA-secure GIBE then* $\mathbf{S}$ *is an adaptively, existentially unforgeable signature scheme. Likewise, if* $\mathbf{G}$ *is a selectively CPA-secure GIBE, then* $\mathbf{S}$ *is a selectively unforgeable signature scheme. In either case,* $\mathbf{G}$ *need not be anonymous.*

Proof. Let $\mathcal{A}$ be an adversary against $\mathbf{S}$. We will construct an adversary $\mathcal{B}$ against $\mathbf{G}$, by simulating $\mathcal{A}$'s interaction with $\mathbf{S}$. By construction, we can translate $\mathcal{A}$'s delegation and signature queries against $\mathbf{S}$ to delegation queries against $\mathbf{G}$. Eventually, $\mathcal{A}$ will output an attempted forgery $(\mathfrak{p}, m, s)$, where s is a purported secret key for a role $(\mathfrak{p}, (\mathfrak{p}, m))$. For this to be a valid forgery, $\mathfrak{p}$ must not be under any role used in a delegation query, and the pair $(\mathfrak{p}, m)$ must not have been used in any signature query.

As a result, the simulation will not have generated any delegation queries for roles above $(\mathfrak{p}, (\mathfrak{p}, m))$.

The simulator $\mathcal{B}$ then challenges $\mathbf{G}$ on the policy $(\mathfrak{p}, m)$ with two different random messages v_0 and v_1, receiving a ciphertext c. It runs

$$v' \leftarrow \textsc{Decrypt}(\text{PP}, (\mathfrak{p}, i), s, (\mathfrak{p}, i), c)$$

If $v' = v_0$, it guesses $b = 0$, and if $v' = v_1$ it guesses $b = 1$; otherwise it guesses at random. The probability of either causal case occurring is $\text{Adv}(\mathcal{A} \leftrightarrow \mathbf{S})$. Since v_0 and v_1 are random, the probability that $v' = v_0$ but that $b = 1$, or vice versa, is negligible (at most $\frac{1}{|\mathcal{M}|-1}$). Thus the probability of a correct guess is $\text{Adv}(\mathcal{A} \leftrightarrow \mathbf{S})$ due to the causal case, and at least $\frac{1}{2}(1 - \text{Adv}(\mathcal{A} \leftrightarrow \mathbf{S}) - \frac{1}{|\mathcal{M}|-1})$ due to a random guess being correct. So

$$\text{Adv}(\mathcal{B} \leftrightarrow \mathbf{G}) \geq \text{Adv}(\mathcal{A} \leftrightarrow \mathbf{S}) - \frac{1}{|\mathcal{M}| - 1}$$

Since the space of possible messages is generally extremely large, $\frac{1}{|\mathcal{M}|-1}$ is almost always negligible. This completes the proof in the adaptive case. For the selective case, note that if $\mathbf{S}$ is given $\mathfrak{p}$ and m ahead of time, it can pass this information to $\mathbf{G}$, so if $\mathcal{A}$ is a selective adversary, then $\mathcal{B}$ is, too. □

Notice that this extra dimension is already present in a CCA_2-secure GIBE constructed according to Theorem 1. Thus a user with a key for a role $\mathfrak{r}$ can sign messages using another role $\mathfrak{r}' \preceq \mathfrak{r}$ as a policy. Furthermore, if the encodings of $(\mathfrak{p}, c)$ and $(\mathfrak{p}, m)$ (or the functions used to hash them when implementing $\mathfrak{I}$) are chosen correctly, the use of this signature will not impact the security of $\mathbf{G}$. Such a system is called an *anonymous ring signature* [30]: it signs a message with proof of a role $\mathfrak{r} \succeq \mathfrak{r}'$ without revealing any information about $\mathfrak{r}$. As in [30], this can be done by taking advantage of an existing cryptosystem, without any additional setup.

Chapter 4

Spatial and Doubly-Spatial Encryption

Our main contribution is a pair of GIBEs, called *spatial encryption* and *doubly-spatial encryption*. The hierarchies for these GIBEs are very rich, but also quite abstract. Their usefulness lies in our ability to embed other GIBEs inside them.

4.1 Spatial Encryption

Affine spaces Let p be a prime number, and let $\mathbb{Z}_p$ done the field of integers modulo p. For some positive integer n, let $\mathbb{Z}_p^n$ denote an n-dimensional vector space over the field $\mathbb{Z}_p$. For any vector $\vec{x} \in \mathbb{Z}_p^n$ and any matrix $M \in \mathbb{Z}_p^{n \times m}$ (i.e. one with n rows and m columns, for some integer $m \geq 0$), we define the *affine subspace* $\mathrm{Aff}\,(M, \vec{x})$ by

$$\mathrm{Aff}\,(M, \vec{x}) \quad := \quad \{\vec{x} + M \cdot \vec{y} : \vec{y} \in \mathbb{Z}_p^m\}$$

If these elements $\vec{x} + M \cdot \vec{y}$ are all unique, we say that $\mathrm{Aff}\,(M, \vec{x})$ is an *m-dimensional* affine subspace. It is a basic theorem from linear algebra that every affine space has a well-defined dimension.

The spatial hierarchy The roles for spatial encryption are all the affine subspaces of $\mathbb{Z}_p^n$. They are ordered by containment: we say that $\mathfrak{r}_1 \succeq \mathfrak{r}_2$ if and only if $\mathfrak{r}_1 \supseteq \mathfrak{r}_2$.

The policies for spatial encryption are all the points of $\mathbb{Z}_p^n$, and we say that $\mathfrak{r} \succeq \mathfrak{p}$ if and only if $\mathfrak{p} \in \mathfrak{r}$. This is the extent of the n-dimensional spatial encryption hierarchy over $\mathbb{Z}_p$, which we call $\mathfrak{S}_p^n$ (but we omit the p most of the time due to laziness). But in the interest of transparently supporting signatures (implemented using Theorem 2), we will define a slight extension of this hierarchy. In the extension, the policies, like the roles, are all the affine subspaces of $\mathbb{Z}_p^n$, with the same ordering. Ordinary spatial encryption embeds into this extension: the points are simply zero-dimensional subspaces.

In order to write down the roles, we will need a compact, unique encoding of each subspace. Just M and $\vec{x}$ will not suffice, because there are many choices of M and $\vec{x}$ that describe the same affine space. Fortunately, linear algebra provides canonical choices of M and $\vec{x}$, along with efficient ways to compute on them. For example, we may require M to be in reduced column-echelon form, and for $\vec{x}$ to be reduced by the columns of M.

In our implementations of spatial encryption, ciphertexts have a constant size regardless of n and regardless of the complexity of the policy (but proportional to the security parameter λ). This will enable us to perform efficient multicast encryption, among other applications. Secret keys have size $O(d \cdot \lambda)$, where d is the dimension of the affine space. This is not as small as we would like, but it is still not enormous.

4.2 Applications of Spatial Encryption

Spatial encryption has many applications, and because it is efficient in terms of computation time and ciphertext size, these applications often have efficiency comparable to solutions specifically tailored to a given problem.

Vanilla IBE Spatial encryption supports vanilla IBE using only one dimension. Each identity i is encoded as the singleton vector $\langle i \rangle$. The PKG's role, $\top$, is encoded as the entirety of $\mathfrak{S}^1$. Here the identities are elements of $\mathbb{Z}_p$, but we can extend the

result to strings (or any other hashable type) using the hashed embedding lemma.

Hierarchical IBE Similarly, we can implement a HIBE hierarchy with maximum depth n using $\mathfrak{S}^n$. Here the path components are elements of $\mathbb{Z}_p^*$. To a role (a, b, c), we associate the affine space of vectors which begin with $\langle a, b, c\rangle^\top$, namely

$$\{\langle a, b, c, x_4, x_5, \ldots, x_n\rangle^\top : x_i \in \mathbb{Z}_p\}$$

Thus the subspace for $\top$ is $\mathbb{Z}_p^n$ (a common feature in GIBEs that embed into spatial). Then to the policy (a, b, c), we associate the vector

$$\langle a, b, c, 0, 0, \ldots, 0\rangle^\top$$

Because 0 is disallowed as a path component, only roles which are prefixes of (a, b, c) decrypt this policy.

Once again, we can use the hashed embedding lemma to make a HIBE which uses strings as path components.

Product and coproduct schemes Spatial encryption naturally supports product schemes: it is clear how to embed $\mathfrak{S}^m \times \mathfrak{S}^n$ into $\mathfrak{S}^{m+n}$. As a result, spatial encryption supports extended coproducts as well, via the extended coproduct embedding lemma. In other words, given several hierarchies which embed into n-dimensional spatial encryption, we can implement their extended coproduct using $n + 1$-dimensional spatial encryption, simply by using the first coordinate to encode which hierarchy we're using. Furthermore, any implementation of spatial encryption can be made CCA_2-secure using Theorem 1, and can be turned into an anonymous ring signature scheme using Theorem 2, in either case at the cost of a single dimension.

Whitelists We can use spatial encryption to implement a GIBE in which the roles and policies are multi-sets of at most n numbers in $\mathbb{Z}_p^*$ (or, via the hashed embedding lemma, in some other set). The policies function as whitelists: a role $\mathfrak{r}$ matches a policy $\mathfrak{p}$ if and only if $\mathfrak{r} \subseteq \mathfrak{p}$, i.e. if each string appearing in $\mathfrak{r}$ appears at least as

many times in $\mathfrak{p}$. We can define delegation in the same way, with $\mathfrak{r}_1 \succeq \mathfrak{r}_2$ if and only if $\mathfrak{r}_1 \subseteq \mathfrak{r}_2$. We denote such a whitelist scheme $\mathfrak{W}_n$

We can implement $\mathfrak{W}_n$ using an $n+1$-dimensional space in a straightforward way. To each policy $\mathfrak{p}$, we associate the polynomial

$$P_{\mathfrak{p}}(t) \ := \ \prod_{x \in \mathfrak{p}} (1 + tx)$$

We encode this polynomial as the $n+1$ coefficients on $t^n, t^{n-1}, \ldots, t, 1$. The polynomial has degree $|\mathfrak{p}| \leq n$ (exactly, because we have disallowed $x = 0$), so this encoding completely describes $P_{\mathfrak{p}}(t)$. Similarly, to each role $\mathfrak{r}$, we associate the polynomial

$$P_{\mathfrak{r}}(t) \ := \ \prod_{x \in \mathfrak{r}} (1 + tx)$$

We encode the role $\mathfrak{r}$ as the subspace of coefficients of all polynomials (with degree at most n) that are divisible by $P_{\mathfrak{r}}(t)$. It is easily seen that this is a vector subspace (i.e. an affine subspace passing through $\vec{0}$) of $\mathbb{Z}_p^{n+1}$: it is closed under addition and under multiplication by elements of $\mathbb{Z}_p$.

Furthermore, notice that all the policies have a final coefficient of 1. Therefore, we can restrict the roles to only those polynomials which are divisible by $P_{\mathfrak{r}}(t)$ and whose final coefficient is 1; these are affine subspaces of $\mathbb{Z}_p^{n+1}$. Of course, once we have done this, we can simply throw away that final coefficient, giving us an embedding into $\mathfrak{S}^n$.

Because $\mathbb{Z}_p[t]$ is a unique factorization domain, $P_{\mathfrak{r}}(t)$ divides $P_{\mathfrak{p}}(t)$ only if $\mathfrak{r}$ is a subset of $\mathfrak{p}$. The same statement is true of pairs of roles, so that this description is indeed a valid embedding.

Multicast IBE Whitelists provide a natural way to implement multicast IBE. Recall that in multicast IBE, the each role is either $\top$ or a single identity, and each policy is a set of identities. A non-$\top$ role decrypts a policy if it is a member of the set. If we restrict policies to have at most n identities, then multicast IBE embeds naturally into whitelist IBE.

For a system to be called "multicast" (or, more traditionally, "broadcast"), its ciphertexts should have size $o(r)$, where r is the number of recipients. In our implementation of spatial encryption, ciphertexts have constant size irrespective of n and r, so this embedding qualifies as a multicast IBE.

Multicast HIBE Whitelists provide an alternate implementation of hierarchical IBE. Consider a whitelist IBE where (via hashed embedding) the identities are sequences of strings. Then we can encode the role or policy for a given path (a, b, c) as the set $\{(a), (a, b), (a, b, c)\}$, that is, the set of nonempty prefixes of the path.

But unlike our other embedding of HIBE, this technique allows multicast. To encode a policy which is a set S of paths, we can simply encode the set of all nonempty prefixes, namely

$$\bigcup_{p \in S} \{\ p' : p' \text{ is a prefix of } p,\ p' \neq ()\ \}$$

When embedded in a space of dimension n, this multicast HIBE allows us to encrypt to any collection of paths containing at most n different prefixes. For example, when applied to email, it is more efficient at sending a message to many users in a single domain than to many users in different domains. Ciphertexts still have a constant size.

This is the first, and to our knowledge, the only existing construction of multicast HIBE.

Flexible and sparse products Whitelists admit efficient constructions of products and coproducts. If $\mathfrak{H}_1$ and $\mathfrak{H}_2$ support embeddings e_1 and e_2 into $\mathfrak{W}_m$ and $\mathfrak{W}_n$, respectively, then their product embeds into $\mathfrak{W}_{m+n}$. We simply set

$$e(\mathfrak{r}_1, \mathfrak{r}_2) = \{(1, x) : x \in e_1(\mathfrak{r}_1)\} \cup \{(2, y) : y \in e_2(\mathfrak{r}_2)\}$$

and likewise for policies. The same formulation of e embeds $e_1 \coprod e_2$ into $\mathfrak{W}_{\max(m,n)}$.

More generally, whitelists support a form of *flexible product*. Consider a hierarchy $\mathfrak{H}$ with an embedding e into $\mathfrak{W}_m$, where $m > n$. Now, some of the roles and policies of $\mathfrak{H}$ may map to lists of length at most n. These define an *n-sparse* sub-hierarchy

$\mathfrak{H}|_n$ of $\mathfrak{H}$ which can be embedded in $\mathfrak{W}_n$. Note that we implicitly used this restriction in our constructions of multicast IBE and HIBE above.

In many cases, the length of $e(\mathfrak{r})$ or $e(\mathfrak{p})$ corresponds to some intuitive notion of the "complexity" of the role or policy. In multicast IBE, for example, $|e(\mathfrak{p})|$ is the number of recipients of a message. Thus, a sparse sub-hierarchy is one which supports limited complexity.

Sparsity is especially useful in product schemes, in which it becomes a limit on the *total* complexity of roles and policies in the product. For each i in some index set I, let $\mathfrak{H}_i$ be a hierarchy supporting an embedding into $\mathfrak{W}_{n_i}$. Then instead of implementing $\prod_{i \in I} \mathfrak{H}_i$ by embedding it into $\mathfrak{W}_{\sum n_i}$, we might instead implement an n-sparse version $(\prod_{i \in I} \mathfrak{H}_i)|_n$ using $\mathfrak{W}_n$ for some $n < \sum n_i$. Note that this can work even if the index set I is enormous, so long as most of the roles and policies in the product map to the empty set.

The same technique has benefits even if we do not wish to create additional restrictions. For example, let $\mathfrak{M}_n$ denote a multicast IBE supporting up to n recipients, and suppose that we wish to implement a product of two multicast IBE schemes. We could embed $\mathfrak{M}_m \times \mathfrak{M}_n$ in $\mathfrak{W}_{m+n}$. But a more flexible design would be to embed $(\mathfrak{M}_{m+n} \times \mathfrak{M}_{m+n})|_{m+n}$ in $\mathfrak{W}_{m+n}$. Instead of allowing the first component to have up to m recipients and the second to have up to n, we would the allow any combination which sums to less than $m + n$.

Spatial sparse products We have seen the utility of sparse products of whitelist encryption schemes; can we accomplish the same for spatial encryption itself? It turns out that we can, but not quite as efficiently. To do this, we will make use of *Vandermonde vectors*. The Vandermonde vector $\vec{v}_x$ is defined as

$$\vec{v}_x := \left\langle 1, x, x^2, \ldots, x^{n-1} \right\rangle^\top$$

These vectors are useful because any n different Vandermonde vectors are linearly independent, and so span $\mathbb{Z}_p^n$. If this were not the case, then there would be some vector $\vec{w}$ orthogonal to all of them. The coefficients of $\vec{w}$ would define a polynomial

$P_{\vec{w}}$ of degree $n-1$, where ${\vec{v}_x}^\top \vec{w} = P_{\vec{w}}(x)$. But then $P_{\vec{w}}$ would have n different roots, one for each of the Vandermonde vectors, which is impossible since its degree is only $n-1$.

Continuing, let $\mathfrak{H}$ be a hierarchy with an embedding e into spatial encryption over $\mathbb{Z}_p^m$ where $m < p$. Further suppose that each policy has at most n nonzero coordinates, and that in each role, all but n coordinates are fixed to 0. Then $\mathfrak{H}$ can be embedded in $\mathbb{Z}_p^{2n}$ by an embedding e'. To do this, let M be the $2n \times m$ matrix whose ith column is $\vec{v}_i$, and set

$$e'(\mathfrak{r}) = M \cdot e(\mathfrak{r}) \quad \text{and} \quad e'(\mathfrak{p}) = M \cdot e(\mathfrak{p})$$

It is clear that if $e(\mathfrak{r}_1) \supseteq e(\mathfrak{r}_2)$ then $e'(\mathfrak{r}_1) \supseteq e'(\mathfrak{r}_2)$. To show the converse, note that together, the points of $e(\mathfrak{a})$ and $e(\mathfrak{b})$ are nonzero on at most $2n$ different coordinates. But any collection of at most $2n$ columns of M are linearly independent, so that the restriction of M to these coordinates is invertible. This establishes the desired converse.

Note that we do not need to actually compute M in order to compute e'; we can instead find the nonzero entries of $e(\mathfrak{p})$ and compute only those columns of M. As a result, m can be extremely large. In fact, using hashed embedding, we can assign a dimension to every string.

More practically, suppose that we have many hierarchies $\mathfrak{H}_i = (\mathfrak{R}_i, \mathfrak{P}_i)$, where i is a description of $\mathfrak{H}_i$. Suppose that there is an embedding $e_i : \mathfrak{H}_i \to \mathbb{Z}_p^{n_i}$, with a *default role* $\hat{\mathfrak{r}}_i$ and a *default policy* $\hat{\mathfrak{p}}_i$, both of which map to $\vec{0}$. Then we can compute a sparse product $\overline{\prod}^n \mathfrak{H}_i$, which embeds into $\mathbb{Z}_p^{2n}$. Its roles are $(\mathfrak{r}_i : \mathfrak{r}_i \in \mathfrak{R}_i)$, subject to the constraint that

$$\sum_{\mathfrak{r}_i \neq \hat{\mathfrak{r}}_i} n_i \leq n$$

and likewise for policies.

In some cases, we might not want he maximum dimensions for roles and policies to be the same. If the total dimension of non-default roles is a, and the total dimension of non-default policies is $b < a$ (a policy with $b > a$ clearly cannot be decrypted

except by $\top$), then we might hope to encode the result in $\mathbb{Z}_p^{a+b}$. In this case, the same argument shows that our embedding preserves the decryption relation, but it might create additional possibilities for delegation. That is, we could have $e(\mathfrak{r}_1) \succeq e(\mathfrak{r}_2)$ even though $\mathfrak{r}_1 \not\succeq \mathfrak{r}_2$, but only in the case that for all (b-sparse) policies $\mathfrak{p} \preceq \mathfrak{r}_2$, we also have $\mathfrak{p} \preceq \mathfrak{r}_1$. Since our definition of security says nothing about delegation, such a system would still be secure, and suitable for encryption. However, the extra delegations would render it unsuitable for ring signatures unless additional precautions are taken. This is expected, because Theorem 2 only applies to GIBEs which have a policy for every role, and this one does not when $b < a$.

Another way that we can reduce the number of dimensions required is if the number of hierarchies in use can be bounded by a parameter $\ell < n/2$, in addition to their total dimension. In this case, we can use a 2ℓ-dimensional sparse product to determine which which hierarchies are in use, and another n-dimensional sparse product to hold the data for those hierarchies.

It should be noted that spatial sparse products are different from whitelist sparse products in one very important way. In a whitelist sparse product scheme, the default role and policy is $\top$. That is, if one of the product hierarchies $\mathfrak{H}_i$ isn't used in a role, that role places no restrictions on the user. Conversely, if $\mathfrak{H}_i$ is not mentioned in a product, roles with a non-default $\mathfrak{H}_i$ component can't decrypt it. Spatial sparse products are nearly the opposite: adding a non-default $\mathfrak{H}_i$ component to a role may give the bearer more power (or less, if that component does not contain $\vec{0}$), and adding such a component to a policy always requires a decryptor to have some non-default $\mathfrak{H}_i$ component in his role.

However, we can accomplish a "default-$\top$" sparse product as well, and we can implement it in $\mathbb{Z}_p^\ell \times \mathbb{Z}_p^n$, where n is the maximum total dimension and ℓ is the maximum number of hierarchies in use. We will first describe what spaces the roles map to. As above, we use the first component as a whitelist of size ℓ to determine which hierarchies are in use. We then consider the second component as coefficients of a degree-$(n-1)$ polynomial Q. For each hierarchy $\mathfrak{H}_i$ of dimension n_i, set a linear

transform $T_i : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n_i}$ as

$$T_i(Q) = \langle Q(H(i,1)), \ldots, Q(H(i,n)) \rangle^\top$$

where H is a collision-resistant hash function. The second component of a role $\mathfrak{r}$ is the affine space of polynomials

$$\{Q : T_i(Q) \in e_i(\mathfrak{r}_i) \text{ for all non-}\top \text{ components } \mathfrak{r}_i \text{ of } \mathfrak{r}\}$$

which we can compute using Lagrange interpolation. The policy then has as its second component a minimal-degree Q satisfying

$$T_i(Q) = e_i(\mathfrak{p}_i) \text{ for all non-}\top \text{ components } \mathfrak{p}_i \text{ of } \mathfrak{p}$$

which once again we can compute by interpolation. It is clear that this is an embedding: if $\mathfrak{H}_i$ is used in $\mathfrak{r}$ but not in $\mathfrak{p}$ (which would mean that $\mathfrak{r} \not\geq \mathfrak{p}$) then the first component prevents decryption. If $\mathfrak{r}_i \not\geq \mathfrak{p}_i$, then the second component prevents decryption. If this is not the case for any $\mathfrak{r}_i$ or $\mathfrak{p}_i$, then interpolation will succeed.

Simple attribute-based encryption Equipped with sparse products, we can build simple variants of attribute-based encryption [21]. For example, we could build the n-sparse product of all concisely describable hierarchies, indexed by a collision-resistant hash of the description. As mentioned above, we need only consider the dimensions which are actually used when encrypting or decrypting.

We can then attach attributes, such as "top-secret clearance", to certain keys, and require them to be set in products.

Forward-secure encryption In a forward-secure encryption system, each key and each message has a *timestamp*. Keys' timestamps can be incremented, and a message can only be decrypted by a key with an earlier timestamp. This prevents an adversary from decrypting old messages using a recently compromised key. Similarly, keys may be given an expiration date, so that an adversary who steals a key unnoticed cannot

use it forever. It is clear that this can be modeled as a GIBE, in which the roles are intervals of timestamps, and the policies are timestamps. Furthermore, by taking a product with this GIBE, we can add forward security to any scheme.

In [13], Canetti et al. show that these designs can be implemented for t timesteps using $O(\log T)$ HIBE keys, and without any ciphertext expansion. We can use their construction with essentially no modification, but we would rather avoid the overhead of having $O(\log T)$ keys. When T is small – say, at most a few hundred timesteps per key – forward security can be embedded in T-dimensional spatial encryption. The key for $[t_1, t_2]$ corresponds to the vector subspace in which $x_i = 0$ for $i < t_1$ and for $i > t_2$, and the policy t maps to the t-th basis vector.

Note that if keys are valid for some maximum period $T' < T$, then it will be more efficient to use a sparse scheme. This can either be a sparse spatial scheme, as above (which takes $T' + 1$ dimensions, or $T' + 2$ if signatures are supported), or a whitelist. The whitelist version works by assigning an identity to each T'-long interval. A secret key for an interval I is written as the intersection of all T'-long intervals that contain I; there are at most T' of them. Similarly, the policy for a timestamp t is written as all T'-long intervals that contain t.

Encrypted email We are now ready to pose a solution to the motivating problem of email encryption from Section 1.4. Recall that we wish to have many private key generators. Each PKG can certify users by their email addresses, or may give a company a key for its domain, which can then be used to certify its users. When sending a message, a user encrypts to the email addresses of the intended recipients, but also chooses which authorities she trusts to certify those recipients.

This is naturally seen as a product between a broadcast HIBE and a broadcast IBE. The broadcast IBE is used for authorities. Each authority a holds a key to $(\top, a)$. The user `bob@microsoft.com`, when certified by authority a, holds a key for $\mathfrak{r}_{\mathrm{bob},a} := ((\texttt{com}, \texttt{microsoft}, \texttt{@}, \texttt{bob}), a)$. Since $(\top, a) \succeq \mathfrak{r}_{\mathrm{bob},a}$, the authority can use DELEGATE to produce this key for Bob. Furthermore, a can certify Microsoft to generate keys for `microsoft.com` email addresses by giving them a key for $((\texttt{com}, \texttt{microsoft}, \texttt{@}), a)$; or, to include subdomains, for $((\texttt{com}, \texttt{microsoft}), a)$.

When Alice encrypts her message, she chooses a set R of recipients and A of authorities, and sets the policy to (R, A). Bob can then decrypt the message if $(\texttt{com}, \texttt{microsoft}, \texttt{@}, \texttt{bob}) \in R$ (or if any extension of it is in R), and also $a \in A$. We can embed this system in $\mathfrak{W}_n$, and thus in spatial encryption of dimension n, subject to the restriction that the "complexity" of the policies — in this case the number of authorities plus the total number of path components in R — is at most n.

Of course, if too many authorities and users are specified, all is not lost. We can divide up the set $R \times A$ into smaller sets, each of which has complexity less than n, and encrypt to each of those sets. The resulting ciphertexts will not be as compact, but the system will still work.

Furthermore, using the designs shown above, we can add other powerful features to our email encryption scheme. For example, we can make keys expire, or make encryption forward-secure, or we can enforce conditions on other attributes to which the PKG itself can assign meaning.

Zero-handshake TLS The same technology could be used to reduce the number of round-trip communications in protocols such as Transport Layer Security (TLS). In current designs, the server sends a public key certificate, which the client uses to encrypt a session key. Instead, the client could use a multi-authority IBE to encrypt the session key, saving a round-trip.

Upper and lower bounds We provide some very rough upper and lower bounds on what hierarchies can be embedded in $\mathfrak{S}_n$. First, if $\mathfrak{H} = (\mathfrak{R}, \mathfrak{P})$ and $|\mathfrak{R} \cup \mathfrak{P}| = n+1$, then $\mathfrak{H}$ can be embedded into $\mathfrak{S}_n$. To do this, assign each role or policy $\mathfrak{a}$ to a different vector $\vec{x}_\mathfrak{a}$ in the set $\{\vec{0}, \vec{e}_1, \ldots, \vec{e}_n\}$ where $\vec{e}_i$ is the ith basis element. Each policy $\mathfrak{p}$ maps to $\vec{x}_\mathfrak{p}$, and each role $\mathfrak{r}$ maps to the smallest subspace containing the points $\vec{x}_\mathfrak{p}$ where $\mathfrak{r} \succeq \mathfrak{p}$ and $\vec{x}_{\mathfrak{r}'}$ where $\mathfrak{r} \succeq \mathfrak{r}'$. This is clearly an embedding. If we do not desire an embedding, but merely a secure implementation, then the points for the roles can be dropped. For most useful hierarchies, the result will be an embedding anyway.

For a first lower bound, note that if a hierarchy contains a chain of n policies, each delegating to the next, then it cannot be embedded in $\mathfrak{S}_m$ where $m < n - 1$,

because each delegation loses a dimension. For a broader result, call a set of policies $S \subseteq \mathfrak{P}$ *general* if for all nonempty $T \subseteq S$, there is a role $\mathfrak{r} \in \mathfrak{R}$ which decrypts all but one of them, and say that a set of points X is in *general position* if all subspaces containing them are at least $(|X| - 1)$-dimensional. The following lemma establishes a lower bound on hierarchies with large general sets.

Lemma. *A secure implementation (and thus an embedding) must map a general set of policies to a set of points in general position.*

Proof. If not, then there would be a minimal subset $T \subseteq S$ which is not in general position. If we were to remove any point $\vec{x}$ from T, then by minimality, the remaining points would be in general position and so would define the same subspace. Thus any role containing $T \backslash \{\vec{x}\}$ would contain $\vec{x}$ as well, and this would be true for all $\vec{x} \in T$. But then by definition, S would not be general, which is a contradiction. □

As a corollary, if in a hierarchy $\mathfrak{H} = (\mathfrak{R}, \mathfrak{P})$ the policy set $\mathfrak{P}$ contains a general set of n policies, then $\mathfrak{H}$ cannot be embedded into $\mathfrak{S}_m$ where $m < n - 1$.

4.3 Doubly-Spatial Encryption

Spatial encryption is powerful, efficient and supports a strong proof of security, but it is not always as expressive as we would like. To improve its expressiveness, we define a variant called *doubly-spatial encryption*.

In doubly-spatial encryption, we allow policies, as well as roles, to be any affine subspace of $\mathbb{Z}_p^n$. We say that $\mathfrak{r} \succeq \mathfrak{p}$ if $\mathfrak{r}$ intersects $\mathfrak{p}$ as subspaces of $\mathbb{Z}_p^n$. We call this hierarchy $\mathfrak{D}_n$.

The reader will note that the roles and policies are not the same, but rather opposites of each other. For example, $\mathbb{Z}_p^n$ is the strongest role, but the weakest policy, and points are the strictest policies, but the most restricted roles. We can, however, extend this definition to allow for "role-like policies" and "policy-like roles". We do this for any role other than points, which already behave the same way for roles and for policies.

Concretely, consider the partially ordered set whose elements are affine subspaces W of $\mathbb{Z}_p^n$, along with a sign $+$ or $-$. For a point space $\{\vec{v}\}$, we say that $+\{\vec{v}\} = -\{\vec{v}\}$. For the partial order, we say that

$$\begin{array}{lcll} +V & \succeq & +W & \text{iff } V \supseteq W \\ -V & \succeq & -W & \text{iff } V \subseteq W \\ +V & \succeq & -W & \text{iff } V \cap W \neq \{\} \\ -V & \succeq & +W & \text{iff } V = W = \{\vec{v}\} \end{array}$$

The roles of doubly-spatial encryption are "positive" spaces (or points) and the policies are "negative" spaces (or points), so when using it, we will write roles and policies with no sign. Our construction supports positive policies and negative roles, but in most cases we find them less useful, and they clutter the notation, so we will leave them as an extension.

Our current construction of doubly-spatial encryption is not as efficient as for spatial encryption. The ciphertexts for a policy W have size $O(\lambda \cdot \dim W)$ rather than $O(\lambda)$, but only one pairing is required so long as the role is positive and the policy negative. More seriously, we have only been able to prove selective security without invoking the generic group model.

4.4 Applications of Doubly-Spatial Encryption

Spatial encryption can be embedded in doubly-spatial encryption. Furthermore, like spatial, doubly-spatial encryption is closed under products and coproducts.

Sparse products with defaults We defined sparse products of hierarchies for spatial encryption. However, we were forced to choose between default roles and policies being $\top$ in a particular hierarchy, or being $\{\vec{0}\}$. With doubly-spatial encryption, we can eliminate this disadvantage.

For all i in some (large but hashable) index set I, let $\mathfrak{H}_i$ be a hierarchy equipped with a default role $\hat{\mathfrak{r}}_i$ and a default policy $\hat{\mathfrak{p}}_i \preceq \hat{\mathfrak{r}}_i$, and suppose that it supports an embedding e_i into $\mathfrak{D}_{n_i}$.

Choose integers ℓ and n, which will serve as limits on the roles and policies we support: we must have

$$|\{\mathfrak{H}_i : \mathfrak{H}_i \text{ used}\}| \leq \ell \quad \text{and} \quad \sum_{\mathfrak{H}_i \text{ used}} n_i \leq n$$

in both roles and policies. We will embed the resulting system in $\mathfrak{D}_{2n+3\ell}$. We will decompose this as a whitelist of length ℓ, embedded in $\mathfrak{S}_\ell$; a simplified attribute-based encryption system supporting ℓ attributes, embedded in $\mathfrak{S}_{2\ell}$; and a doubly-spatial system $\mathfrak{D}_{2n}$.

We will use the whitelist to prevent decryption when a policy $\mathfrak{p}$ does not use a hierarchy $\mathfrak{H}_i$ (and is thus equal to the default policy in that hierarchy), but $\mathfrak{r}_i$ cannot decrypt that default policy. Given a role $\mathfrak{r}$, consider each component $\mathfrak{r}_i$ which does not intersect the default policy $\hat{\mathfrak{p}}_i$ of its hierarchy. To ensure that $\mathfrak{r}$ cannot decrypt a policy $\mathfrak{p}$ unless it mentions hierarchy i, we put i on the whitelist for $\mathfrak{r}$. Similarly, for each component $\mathfrak{p}_i$ of $\mathfrak{p}$ which is not contained in $\hat{\mathfrak{p}}_i$, we put i on the whitelist for $\mathfrak{p}$. Notice that this construction respects delegation, because if $\mathfrak{r}_i$ does not intersect $\hat{\mathfrak{p}}_i$, then neither do any roles that can be delegated from it.

In the same way, we will use the attribute-based system to prevent decryption when a role $\mathfrak{r}$ does not use $\mathfrak{H}_i$, but $\mathfrak{p}$ does and $\mathfrak{p}_i$ is not decryptable by its default role $\hat{\mathfrak{r}}_i$. To do this, give $\mathfrak{r}$ access to the attribute i when $\mathfrak{r}_i$ is not contained in $\hat{\mathfrak{r}}_i$, and have the policy require that attribute when $\mathfrak{p}_i$ does not intersect $\hat{\mathfrak{r}}_i$.

Finally we come to the doubly-spatial system $\mathfrak{D}_{2n}$. This we use as in the default-$\top$ sparse spatial products. Choose a collision-resistant hash function $H : I \times [1, n] \to \mathbb{Z}_p$, and interpret each element of $\mathbb{Z}_p^{2n}$ as a polynomial P of degree at most $2n - 1$. For each non-default role component $\mathfrak{r}_i$, consider the linear transform $T_i : \mathbb{Z}_p^{2n} \to \mathbb{Z}_p^{n_i}$ defined by

$$T_i(P) = \langle P(H(i, 1)), \ldots, P(H(i, n_i)) \rangle^\top$$

Then define the role's subspace

$$W_\mathfrak{r} := \{P \in \mathbb{Z}_p^{2n} : \text{for all } i \text{ such that } \mathfrak{r}_i \neq \hat{\mathfrak{r}}_i,\ T_i(P) \in e_i(\mathfrak{r}_i)\}$$

and likewise for policies. Since the constraints are all affine, these are clearly affine subspaces of $\mathbb{Z}_p^{2n}$. Delegation clearly works; to see that decryption works, note that the intersection of $W_{\mathfrak{r}}$ with $W_{\mathfrak{p}}$ contains the space

$$\{P \in \mathbb{Z}_p^{2n} : \text{for all } i \text{ such that } \mathfrak{r}_i \neq \hat{\mathfrak{r}}_i \text{ or } \mathfrak{p}_i \neq \hat{\mathfrak{p}}_i,\ T_i(P) \in e_i(\mathfrak{r}_i) \cap e_i(\mathfrak{p}_i)\}$$

If $e_i(\mathfrak{r}_i)$ and $e_i(\mathfrak{p}_i)$ intersect in some set containing a point $\vec{x}$, then barring a hash collision, we can construct a preimage of $\vec{x}$: we must simply find a polynomial P of degree at most $2n-1$ with a given value at $2n$ or fewer distinct points (n for the role and n for the policy), and we can find it easily by Lagrange interpolation. However, if $\mathfrak{r}_i$ and $\mathfrak{p}_i$ do not intersect, and both are specified (rather than being the default role or policy), then the constraint that $T_i(P)$ be in both $e_i(\mathfrak{r}_i)$ and $e_i(\mathfrak{p}_i)$ is unsatisfiable. If they do not intersect and one is the default, then the whitelist or the simple ABE scheme will prevent decryption.

We can save a small amount of space in the secret keys and ciphertexts by choosing n dummy points which cannot be the output of the hash function H. Then in each role or policy whose n_i sum to $n' < n$, we can specify that the first $n - n'$ dummy points are roots of the polynomial P, so that the $\mathfrak{D}_{2n}$ component has dimension exactly n.

It is easily seen that we cannot securely implement a comparable system using spatial encryption, even if we restrict all policies to be points. Consider a product of d copies of $\mathfrak{S}_1$ with default role $\top$ and default policy 0, and suppose that we allow only one component to differ from the default. Let $\mathfrak{p}_i$ be 1 in the ith position and 0 elsewhere. Let $\mathfrak{r}_i$ be $\{0\}$ in the ith position and $\top$ elsewhere. Then $\mathfrak{r}_i \succeq \mathfrak{p}_j$ if and only if $i \neq j$, so that the d policies $\{\mathfrak{p}_i\}$ are general. Thus this system cannot be securely implemented in $\mathfrak{S}_{d-2}$ or less, whereas this application embeds it in $\mathfrak{D}_5$.

Wide intervals We explained how to encode intervals into sparse spatial products, so long as the intervals were guaranteed to be $O(n)$ in width. We now explain how to encode wider intervals. Of course, we can no longer support arbitrary delegation, because even with doubly-spatial encryption no delegation chain can be more than

$n+1$ roles long. In particular, we want to make a flat poset of roles, one for each integer in $[0,k)$ for some k, plus $\top$. Policies will also be integers in $[0,k)$, and we want $\mathfrak{r} \succeq \mathfrak{p}$ if and only if $\mathfrak{r} \geq \mathfrak{p}$.

To do this, fix an integer d. We will write integers in $[0,k)$ as d-digit integers in base $b := \lceil k^{1/d} \rceil$, using up to $((b-1)d)$-dimensional roles and $(d-1)$-dimensional policies. For the roles, partition $\mathbb{Z}_p^{(b-1)d}$ into d copies of $\mathbb{Z}_p^{b-1}$, and let $S_i \subseteq \mathbb{Z}_p^{b-1}$ be the vector subspace spanned by the first i basis vectors of $\mathbb{Z}_p^{b-1}$ (so that $S_0 = \{\vec{0}\}$ and $S_{b-1} = \mathbb{Z}_p^{b-1}$). Then to the role whose digits are $(r_0, r_1, \ldots, r_{d-1})$ with r_{d-1} being the most significant digit, we assign the subspace

$$S_{r_0} \times S_{r_1} \times \ldots \times S_{r_{d-1}}$$

For the policies, let $\vec{e}_i$ be the ith basis vector of $\mathbb{Z}_p^{b-1}$ when $i > 0$, and let $\vec{e}_0 = \vec{0}$; thus $\vec{e}_i \in S_i$. To the policy whose digits are $(p_0, p_1, \ldots, p_{d-1})$, we assign the $(d-1)$-dimensional subspace which passes through the d points

$$(\vec{e}_{p_0}, \vec{e}_{p_1}, \ldots, \vec{e}_{p_{d-1}}), \quad (\vec{e}_0, \vec{e}_{p_1+1}, \ldots, \vec{e}_{p_{d-1}}), \quad \ldots, \quad (\vec{e}_0, \vec{e}_0, \ldots, \vec{e}_0, \vec{e}_{p_{d-1}+1})$$

If any of the p_i is equal to $b-1$ then we omit the ith vector, so the subspace may actually have dimension less than $d-1$.

Suppose that $\mathfrak{r} \succeq \mathfrak{p}$. Then either $\mathfrak{r} = \mathfrak{p}$, or $\mathfrak{r}$ is greater than $\mathfrak{p}$ in the ith digit and equal in every higher digit. In either case, decryption will succeed.

Suppose on the contrary that $\mathfrak{r} \not\succeq \mathfrak{p}$. Then in particular $r_{d-1} \leq p_{d-1}$, so that $\vec{e}_{p_{d-1}+1} \notin S_{d_{d-1}}$. Since $\vec{e}_{p_{d-1}+1}$ does not appear in the last component in any of the other defining points of the policy, this last vector is useless. The rest of the points all have $\vec{e}_{p_{d-1}}$ as their last coordinate, so if $r_{d-1} < p_{d-1}$ we certainly cannot decrypt. But if $r_{d-1} = p_{d-1}$, we can repeat this argument on the less significant digits, and we will eventually have $r_i < p_i$ because $\mathfrak{r} < \mathfrak{p}$.

Negated spatial encryption Attrapadung and Libert propose a *negated spatial encryption* [3]. Here roles are affine subspaces of $\mathbb{Z}_p^n$, the policies are points and $\mathfrak{r} \succeq \mathfrak{p}$ precisely when $\mathfrak{p} \notin \mathfrak{r}$. The authors construct a special case, which we will

translate roughly as follows: the subspace W for a role $\mathfrak{r}$ must have $n-1$ dimensions, must pass through $\vec{0}$ (i.e. it must be a vector subspace rather than an affine one), and must not contain $\langle 1, 0, \ldots, 0\rangle^\top$. To construct this special case with constant-size ciphertexts, we give out keys for roles W as normal, and map the policy $\vec{v}$ to the 1-dimensional affine space $\langle 1, 0, \ldots, 0\rangle^\top + \text{span}(\vec{v})$. Then if $\vec{v} \in W$, we know that $\langle 1, 0, \ldots, 0\rangle^\top \notin W = \text{span}(\vec{v}, W)$, so decryption is impossible. On the other hand, if $\vec{v} \notin W$, then together they span the entirety of $\mathbb{Z}_p^n$, so decryption is possible.

As [3] points out, the roles in our identity-based multicast encryption (without the trick that reduces the dimension by 1) have dimension one less than the ambient space and pass through $\vec{0}$ but not through $\langle 1, 0, \ldots, 0\rangle^\top$. Therefore we can negate that system to build identity-based broadcast encryption with constant-size ciphertext.

If we are willing to sacrifice constant-size ciphertext, we can build the entirety of negated spatial encryption. We will retain the restriction that roles must pass through $\vec{0}$, but this can be removed by adding another dimension. We map the role for a subspace W to its orthogonal complement $W^\perp$, and a policy $\vec{v}$ to $\vec{x} + \vec{v}^\perp$, where $\vec{x}$ is any vector not contained in the $(n-1)$-dimensional space $\vec{v}^\perp$. This design allows delegation from a role for W to a role for any $W' \supseteq W$. For decryption, if $\vec{v} \in W$ we will have $W^\perp \subseteq \vec{v}^\perp$, so that $\vec{x} \notin \text{span}(\vec{v}^\perp, W^\perp) = \vec{v}^\perp$. Conversely, if $\vec{v} \notin W$ we will have $W^\perp \not\subseteq \vec{v}^\perp$, so that together they span the entirety of $\mathbb{Z}_p^n$.

Monotone span programs Given a set S and a field $\mathbb{F}$, a monotone span program is a map m from $S \to \mathbb{F}^n$ along with a vector $\vec{v} \in \mathbb{F}^n$. It is interpreted as a function from 2^S to $\{0, 1\}$ as follows: a subset $T \subseteq S$ is accepted if

$$\vec{v} \in \text{span } \{m(t) : t \in T\}$$

Goyal et al. show [21] that attribute-based encryption for monotone boolean formulas can be reduced to monotone span programs. Therefore, it can be implemented using spatial encryption over $\mathbb{F}^{|S|} \times \mathbb{F}^n$. The role for a subset $T \subseteq S$ is encoded as

$$\text{span } \{(\vec{e}_t, 0) : t \in T\}$$

where $\vec{e}_t$ is the tth basis vector of $\mathbb{F}^{|S|}$. The policy for a monotone span program is

$$(\vec{0}, \vec{v}) + \text{span}\ \{(\vec{e}_s, m(s)) : s \in S\}$$

The $\vec{e}_s$ component only appears once in the policy and at most once in the role, so its coefficients must be opposite in those places. Thus any multiple of $m(s)$ can appear if $s \in T$, but no multiple of it can appear if $s \notin T$.

By using sparse products in place of $\mathbb{F}^{|S|}$ we can support sets of arbitrary size, as long as at most n elements are used in any role or policy.

Thresholds Although threshold gates are supported in [21], we will provide an explicit instantiation in doubly spatial encryption with a fixed maximum set size of n. Define the *degree-d complete homogeneous function* on variables $S := \{x_1, \ldots, x_\ell\} \subset \mathbb{Z}_p^*$ by

$$h_d(S) \leftarrow \sum_{i_1 \leq \ldots \leq i_d} x_{i_1} x_{i_2} \cdots x_{i_d}$$

and note that for any $x \in S$ and $d > 0$ we have

$$h_d(S) = h_d(S \backslash \{x\}) + x \cdot h_{d-1}(S)$$

Likewise, define the vector

$$\vec{h}_d(S) \leftarrow \langle h_d(S), h_{d+1}(S), \ldots, h_{d+2n-1}(S) \rangle^\top \in \mathbb{Z}_p^{2n}$$

so that for any $x \in S$ and $d > 0$,

$$\vec{h}_d(S) = \vec{h}_d(S \backslash \{x\}) + x \cdot \vec{h}_{d-1}(S)$$

and $\vec{h}_d(\{x\}) = x^d \cdot \vec{v}_x$ is a multiple of the xth Vandermonde vector. From this recursion, we can efficiently compute $\vec{h}_d(S)$. For $1 \leq d \leq |S|$, define the $(|S| - d)$-dimensional subspace[1]

$$W_{d,S} \leftarrow \vec{h}_{|S|-d}(S) + \text{span } \{\vec{h}_{|S|-d-1}(S), \ldots, \vec{h}_0(S)\}$$

and note that

$$W_{d-1,S} \supset W_{d,S} \supset W_{d,S\setminus\{x\}} \quad \text{and} \quad W_{1,\{x\}} = \text{span}(\vec{v}_x)$$

Likewise, define the $|S|$-dimensional subspace

$$U_S \leftarrow \text{span } \{\vec{v}_x : x \in s\}$$

and note that $\vec{h}_d(S) \in U_S$ for all positive d.

We propose that the subspaces U_S and $W_{d,T}$ can be used to implement a threshold gate. Thus, we may set the roles to be sets S, and the policies pairs (d, T), with decryption allowed only if $|S \cap T| \geq d$. This allows delegation from S to its subsets. Or, we can reverse roles and policies, with delegation allowed from (d, T) to $(d+1, T)$ and to $(d, T\setminus\{x\})$ but not, as [21] allows, to $(d + 1, T \cup \{x\})$.

Furthermore, because this scheme is naturally embedded in a sparse product, we can replace the pairs (d, T) with collections of pairs $\{(d_i, T_i)\}$, where the T_i are disjoint and $\sum |T_i| \leq n$; here decryption is possible if and only if $|S \cap T_i| \geq d_i$ for all i.

Claim. *For subsets $S, T \subset \mathbb{Z}_p^*$ with $|S|, |T| \leq n$, we have $U_S \cap W_{d,T} \neq \{\}$ if and only if $|S \cap T| \geq d$.*

Proof. We will handle the "if" direction first. We know that for all $x \in T$ we must have $W_{1,T} \supseteq W_{1,\{x\}} = \text{span}(\vec{v}_x)$, so that $W_{1,T}$ is the unique $(|T| - 1)$-dimensional subspace passing through these $|T|$ Vandermonde vectors, and $W_{d,T} \subseteq W_{1,T} \subset U_T$. Likewise

$$U_S \supseteq U_{S\cap T} \supset W_{d,S\cap T} \subseteq W_{d,T}$$

[1]From this definition, $W_{d,S}$ has at most $|S| - d$ dimensions, but it is easily seen that the bound is exact.

so that $U_S \cap W_{d,T}$ is nonempty.

Conversely, suppose that $U_S \cap W_{d,T} \neq \{\}$ and let $W \leftarrow \text{span}(U_S, W_{d,T})$ be a vector subspace of $\mathbb{Z}_p^{2n}$. We will show that

$$\vec{v}_x \in W \text{ for all } x \in S \cup T$$

These $|S \cup T|$ vectors are linearly independent because $|S \cup T| \leq 2n$. Furthermore, W is the span of $|S| + |T| - d + 1$ vectors: $|S|$ from U_S, another $|T| - d$ from the vector component of $W_{d,T}$ and another 1 from its affine component. There is at least one linear dependence due to the intersection, so that $\dim W \leq |S| + |T| - d$. Thus we will have

$$|S \cap T| \;=\; |S| + |T| - |S \cup T| \;\geq\; |S| + |T| - (|S| + |T| - d) \;=\; d$$

as desired. Certainly if $x \in S$ then $\vec{v}_x \in W$, so it remains to show this when $x \in T \backslash S$. In that case, we know that

$$W \supset W_{d,T} \supseteq W_{d,T \cap S \cup \{x\}} \ni \vec{h}_d(T \cap S \cup \{x\})$$

Applying the above recursive formulas to $\vec{h}_d(T \cap S \cup \{x\})$, we see that

$$\vec{h}_{d'}(T \cap S \cup \{x\}) \in \text{span}(\vec{h}_d(T \cap S \cup \{x\}),\ U_{T \cap S})$$

for all positive d', so that

$$x \cdot \vec{v}_x \in W_{1,T \cap S \cup \{x\}} \subset W$$

and $\vec{v}_x \in W$ as desired. This completes the proof. □

Although this example is somewhat complex, it also demonstrates the utility of doubly-spatial encryption: even a relatively complex construction requires only a few pages of linear algebra to specify and to prove secure.

All-but-one signatures We will show how to make a signature scheme in which secret keys can sign any message except for one. We will use extended doubly-spatial encryption for this. Messages are encoded in the hierarchy of this signature scheme, so we won't need the extra $\mathfrak{I}$ component used in Theorem 2.

The messages in our scheme will be points of $\mathbb{Z}_p^n$, and we will use extended $(n+2)$-dimensional doubly spatial encryption. Let $H : \mathbb{Z}_p^n \to \mathbb{Z}_p$ be a collision-resistant hash function.

The roles in our scheme will be of the form W or $(W, \vec{v})$, where W is an affine subspace of $\mathbb{Z}_p^n$ and $\vec{v}$ is a point in $\mathbb{Z}_p^n$. The former role allows signatures of any point in W, and the latter allows signatures of any point in W except for $\vec{v}$ (which, formally speaking, need not be in W). We say that $W \succeq W'$ and $W \succeq (W', \vec{v})$ if and only if $W \supseteq W'$. We also allow delegation once a point has been removed, such that $(W, \vec{v}) \succeq (W', \vec{v})$ when $W \supseteq W'$. But the removal cannot be undone (and replaced with a different point $\vec{v}'$) even when $\vec{v} \notin W'$.

The key for a role W is a secret key for

$$+(W \times \mathbb{Z}_p^2) \subseteq \mathbb{Z}_p^{n+2}$$

The key for $(W, \vec{v})$ is a secret key for

$$+(W \times \{(t, t \cdot H(\vec{v})) : t \in \mathbb{Z}_p\})$$

The signature on a message $\vec{v}$ is a secret key for the role

$$-\{(\vec{v}, t, t \cdot H(\vec{v}) + 1) : t \in \mathbb{Z}_p\} \subset \mathbb{Z}_p^{n+2}$$

which is a (negative) one-dimensional subspace. It is clear that this subspace does not intersect the one for any role with $\vec{v}$ removed, but otherwise the two lines in the $\mathbb{Z}_p^n$ component intersect unless a hash collision has occurred. Signatures can be checked using a test decryption with this subspace as the policy. By the same argument as Theorem 2, if extended doubly-spatial encryption is selectively secure then this

signature scheme is selectively unforgeable. Furthermore, because delegation is path-independent, a signature on $\vec{v}$ does not reveal anything about the role used to sign it. We will also see that the signatures are constant-size and can be verified quickly.

To see why this is useful, we consider an application inspired by secure DNS. Suppose that an administrator sets up a secure DNS server, and wishes for all its responses to be signed. Furthermore, the administrator wants to prevent the server from signing false responses *even if it is compromised*. To prevent replay attacks, the responses must be signed in a way that reflects the domain which was queried. Because the server knows about only a limited number of domains, its administrator can simply sign the information about each domain and upload this signature to the server, which echos it when sending responses. This is how DNSSEC works.

But what if a resolver queries a domain name that the server does not know about? There are infinitely many such names, so the administrator cannot pre-sign all possible negative responses. The first proposal, NSEC [33], signs intervals in which there are no positive responses. That is, if domains d_1 and d_2 are adjacent under some ordering, then the administrator gives the server a signature on (d_1, d_2). For all d' where $d_1 < d' < d_2$, this signature will suffice to convince a resolver that the server knows nothing about d'. However, this process reveals d_1 and d_2, so that an adversary can discover all m domains for which the server has information using only $m + 1$ queries. While DNS is usually a public directory, many administrators do not consider such an *unauthorized zone transfer* acceptable. To mitigate this problem, a variant called NSEC3 [24] was proposed which hashes the domains first, using multiple hash functions to increase the number of queries required. But as Bernstein pointed out [4], the domain names can usually be recovered with a dictionary attack.

Suppose that we do not want responses to reveal any information about which domains are present on the server (other than the one which was queried). This already holds for positive responses, so we need only consider negative ones. The server could use zero-knowledge sets [27], but these require significant computation time and bandwidth for each signature. Zero-knowledge sets provide a property that we do not need or even want, namely that the administrator cannot change the server's data without being noticed. Because they do not have this property, our all-but-one

signatures can be more efficient.

Let $H' : \{0,1\}^* \to \{0,1\}^n$ be another collision-resistant hash function (where $n \geq 2\lambda$ for security), and hash each of the m domains on the server with this H'.[2] Consider a binary decision digram (i.e. a trie) for the resulting hashes. It will have m leaves, each containing exactly one domain d. On the path to each leaf, we will have checked several bits B of the hashed domains, some set to 0 and others to 1. Thus the set of domains which lead to each leaf d (those whose B-bits are the same as those of d) form an affine subspace W_d of $\mathbb{Z}_p^n$. For each d, the administrator gives the server a signing key for (W_d, d). Now,

$$\bigcup_{d \in \text{domains}} W_d = \mathbb{Z}_p^n$$

so with these m keys, the server can sign a negative response to any query for an unknown domain. However, since the W_d's do not overlap, the server (or an adversary who compromises it) cannot sign a negative response to a query for any known domain. As we mentioned, the signatures are uniformly random among all valid signatures, and so do not leak any information.

This system is only selectively secure in the standard model. However, as is usually the case for signature schemes, we can recover adaptive security by modeling H' as a random oracle.

[2] We assume that $m > 0$. Otherwise every query will be answered in the negative, so we can just give the server a key for $\top$.

Chapter 5

Construction

5.1 Preliminaries

Before we show how to construct spatial and doubly spatial encryption, we will need some preliminaries on groups.

Recall that our groups are abelian, and written additively, so that the group operation is + and "exponentiation" is written $x \cdot \alpha$; we therefore call it *scalar multiplication* instead of exponentiation.

The *exponent* of a group $\mathcal{G}$ is the smallest positive integer e such that $e \cdot \alpha = 0$ for all $\alpha \in \mathcal{G}$. Thus the operation $x \to x \cdot \alpha$ makes sense as an operation from $\mathbb{Z}_e \times \mathcal{G}$ to $\mathcal{G}$, where e is the exponent of $\mathcal{G}$. Lagrange's theorem states that e divides $|\mathcal{G}|$, so this also makes sense as an operation from $\mathbb{Z}_{|\mathcal{G}|} \times \mathcal{G}$ to $\mathcal{G}$.

Notice that if the exponent e is prime (which we will now call p) then $\mathbb{Z}_p$ is a field, and this operation makes $\mathcal{G}$ (and thus $\mathcal{G}^k$ for any k) a vector space over $\mathbb{Z}_p$. Therefore the order of $\mathcal{G}$ is p^d for some integer d. Conversely, any nontrivial vector space over $\mathbb{Z}_p$ is a group of exponent p. We can also define an "inner product" $\mathbb{Z}_p^k \times \mathcal{G}^k \to \mathcal{G}$ by

$$\vec{n}^\top \vec{\alpha} := n_1 \cdot \alpha_1 + n_2 \cdot \alpha_2 + \ldots + n_k \cdot \alpha_k$$

and, likewise, the product of a matrix over $\mathbb{Z}_p$ with a vector or matrix over $\mathcal{G}$.

While it is convenient for linear algebra, our additive notation is deceptive in one

very important respect. Let $\beta = n \cdot \alpha$. Recovering α from n and β ("dividing by n") is easy if n invertible modulo the exponent e (and otherwise the answer is not well-defined): we simply multiply by n^{-1} mod e. However, recovering n from α and β ("dividing by α") may be very difficult; this is known as the *discrete logarithm problem* from the fact that it looks like a logarithm when groups are written multiplicatively.

Randomized groups The operations in a group are deterministic by definition, but the algorithms to carry them out need not be. In particular, it might be that each group element has more than one possible encoding in the computer's memory, and that group operations output a random encoding of their results. In this case, we call it a *randomized group*, as opposed to a *deterministic group*.

For example, suppose that $\mathcal{G}$ has a subgroup $\mathcal{H}$; then the quotient group $\mathcal{G}/\mathcal{H}$ is the group of equivalence classes of elements of $\mathcal{G}$, where two elements α and β of $\mathcal{G}$ are considered equivalent if $\alpha - \beta \in \mathcal{H}$. We might choose to encode an element of $\mathcal{G}/\mathcal{H}$ by a random element of this equivalence class. After each group operation, we could add a random element of $\mathcal{H}$, so that the result would be a uniformly random representative of its class, independent of the input elements. We call the resulting group $\mathcal{G}/_{\mathrm{R}}\mathcal{H}$. In this case, comparing two group elements for equality might be infeasible even if the group operations are efficient.

Bilinear maps A map $\otimes$ from $\mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$, where $\mathcal{G}_1, \mathcal{G}_2$ and $\mathcal{G}_T$ are groups, is called *bilinear* if

$$(\alpha + \beta) \otimes \gamma = \alpha \otimes \gamma + \beta \otimes \gamma \quad \text{and} \quad \alpha \otimes (\gamma + \delta) = \alpha \otimes \gamma + \alpha \otimes \delta$$

for all $\alpha, \beta \in \mathcal{G}_1$ and $\gamma, \delta \in \mathcal{G}_2$. As a corollary, if $\otimes$ is bilinear, then

$$(n \cdot \alpha) \otimes \gamma \;=\; \alpha \otimes (n \cdot \gamma) \;=\; n \cdot (\alpha \otimes \gamma)$$

for all $\alpha \in \mathcal{G}_1$ and $\gamma \in \mathcal{G}_2$, and all integers n. Here $\mathcal{G}_1$ and $\mathcal{G}_2$ are called the *source groups*, and $\mathcal{G}_T$ is called the *target group*. We will use one of the source groups for keys and the other for ciphertexts, so we will call them $\mathcal{G}_{\mathrm{key}}$ and $\mathcal{G}_{\mathrm{ct}}$ instead of $\mathcal{G}_1$

and $\mathcal{G}_2$. These groups may be the same or different, depending on implementation considerations and how strong a security proof is desired, so we will be careful to make a distinction between them.

There are some simple examples of bilinear maps: the *degenerate* map which takes every element to 0 is bilinear, as are scalar products and inner products. These maps are easily reversible in some directions but not in others, as previously mentioned. However, in this paper, we will use more complex maps which are difficult to reverse on both sides. For readers familiar with elliptic curves, we can use the Weil pairing, or other pairings such as the Tate, η_t or R-ate pairings. Here $\mathcal{G}_{\text{key}}$ and $\mathcal{G}_{\text{ct}}$ are groups of points on certain elliptic curves, and $\mathcal{G}_T$ is the multiplicative group of roots of unity in $\mathbb{Z}_{p^k}$ for some k. Despite this, we will still write $\mathcal{G}_T$ additively.

Affine matrices The n-dimensional affine space $\text{Aff}\left(\mathbb{Z}_p^n\right)$ is formally defined as the subset of $\mathbb{Z}_p^{n+1}$ whose first coordinate is 1. This definition affords a convenient description of affine spaces. It is well-known that a d-dimensional vector subspace of $\mathbb{Z}_p^n$ can be described as

$$\text{span}(M) := \{M \cdot \vec{x} : \vec{x} \in \mathbb{Z}_p^d\}$$

and that $\text{span}(M) \subseteq \text{span}(L)$ if and only if there is a matrix K such that $M = LK$. Likewise, an affine subspace of $\text{Aff}\left(\mathbb{Z}_p^n\right)$ can be described as

$$\text{affspan}(M) := \left\{M \cdot \vec{x} : \vec{x} \in \text{Aff}\left(\mathbb{Z}_p^d\right)\right\}$$

where M has the form $\begin{pmatrix} 1 & \\ \vec{y} & M' \end{pmatrix}$. We call a matrix of this form an *affine matrix*, and note that its dimensions are $(n+1)\times(d+1)$. It is easy to show that $\text{affspan}(M) \subseteq \text{affspan}(L)$ if and only if there is an affine matrix K such that $M = LK$, and that such a K is easy to compute given M and L.

5.2 Standard Constructions

We are now ready to describe the construction of spatial and doubly-spatial encryption. The construction for spatial encryption will emerge as a special case of the construction for doubly-spatial encryption.

GroupGen In order to generate suitable groups for spatial encryption, we assume that we have an efficient, randomized algorithm GroupGen. Given a security parameter λ, this algorithm computes descriptions of groups $\mathcal{G}_{\text{key}}$ and $\mathcal{G}_{\text{ct}}$ of prime exponent p with non-zero elements α_{key} and α_{ct}, respectively, and of a bilinear map $\otimes : \mathcal{G}_{\text{key}} \times \mathcal{G}_{\text{ct}} \to \mathcal{G}_T$ such that $\alpha_{\text{key}} \otimes \alpha_{\text{ct}} \neq 0$. It outputs

$$(p;\ \mathcal{G}_{\text{key}}, \mathcal{G}_{\text{ct}}, \mathcal{G}_T;\ \otimes; \alpha_{\text{ct}}, \alpha_{\text{key}})$$

Typically $\mathcal{G}_{\text{key}}$ and $\mathcal{G}_{\text{ct}}$ will be cyclic of order p and generated by α_{key} and α_{ct}, respectively. However, our proof of adaptive security uses non-cyclic, randomized groups $\mathcal{G}_{\text{key}}$ and $\mathcal{G}_{\text{ct}}$.

Setup Setup is given parameters λ and n. It first runs

$$(p;\ \mathcal{G}_{\text{key}}, \mathcal{G}_{\text{ct}}, \mathcal{G}_T;\ \otimes;\ \alpha_{\text{ct}}, \alpha_{\text{key}}) \leftarrow \text{GroupGen}(\lambda)$$

and sets the *group parameters*

$$\text{GP} \leftarrow (\lambda;\ p;\ \mathcal{G}_{\text{ct}}, \mathcal{G}_{\text{key}}, \mathcal{G}_T;\ \otimes)$$

Setup chooses a *blinding vector*

$$\vec{r} \xleftarrow{\text{R}} \mathbb{Z}_p^{n+1}$$

and sets

$$\vec{\gamma}_{\text{key}} \leftarrow \vec{r} \cdot \alpha_{\text{key}} \in \mathcal{G}_{\text{key}}^{n+1} \quad \text{and} \quad \vec{\gamma}_{\text{ct}} \leftarrow \vec{r} \cdot \alpha_{\text{ct}} \in \mathcal{G}_{\text{ct}}^{n+1}$$

Note that if $\mathcal{G}_{\text{key}} = \mathcal{G}_{\text{ct}}$ is cyclic and $\alpha_{\text{key}} = \alpha_{\text{ct}}$, then $\vec{\gamma}_{\text{key}} = \vec{\gamma}_{\text{ct}}$ will simply be a random vector of group elements. SETUP publishes the *key-blinding parameters*

$$\text{KBP} \leftarrow (\alpha_{\text{key}}, \vec{\gamma}_{\text{key}})$$

It proceeds to choose a *master secret*

$$r_\delta \xleftarrow{\text{R}} \mathbb{Z}_p^*; \qquad \delta_{\text{key}} \leftarrow r_\delta \cdot \alpha_{\text{key}}$$

It computes the *encryption parameters*:

$$\begin{aligned} \tau &\leftarrow \delta_{\text{key}} \otimes \alpha_{\text{ct}} \\ \text{EP} &\leftarrow (\alpha_{\text{ct}},\ \vec{\gamma}_{\text{ct}},\ \tau) \end{aligned}$$

The public parameters are then

$$\text{PP} \leftarrow (\text{GP}, \text{EP}, \text{KBP})$$

Before describing the master secret key, we will describe what a secret key looks like. Each role W is an affine subspace of $\mathbb{Z}_p^n$, of some dimension d, so we can write it as affspan(M) for some affine matrix $M \in \mathbb{Z}_p^{(n+1)\times(d+1)}$. As noted before, we require M to be chosen in some canonical way for each W, so that all secret keys for W will have the same distribution. A secret key for W is then of the form

$$(r \cdot \alpha_{\text{key}}, \quad r \cdot M^\top \cdot \vec{\gamma}_{\text{key}} + \langle \delta_{\text{key}}, 0, \ldots, 0 \rangle^\top) \in \mathcal{G}_{\text{key}}^{d+2}$$

for some random $r \xleftarrow{\text{R}} \mathbb{Z}_p$. In particular, for the master key we may choose $M = \text{Id}_{n+1}$, so that SETUP can compute

$$\begin{aligned} r &\xleftarrow{\text{R}} \mathbb{Z}_p \\ \text{MSK} &\leftarrow (r \cdot \alpha_{\text{key}}, \quad r \cdot \vec{\gamma}_{\text{key}} + \langle \delta_{\text{key}}, 0, \ldots, 0 \rangle^\top) \end{aligned}$$

Finally, SETUP outputs (PP, MSK) as required.

Delegate The algorithm DELEGATE($\mathrm{PP}, \mathfrak{r}_1, \mathrm{SK}_{\mathfrak{r}_1}, \mathfrak{r}_2$) should return a uniformly random secret key for $\mathfrak{r}_2$. Here $\mathfrak{r}_1$ and $\mathfrak{r}_2$ should be given as affine spans of matrices M_1 and M_2, respectively, so we can compute an affine matrix K such that $M_2 = M_1 K$. Then

$$\mathrm{SK}_{\mathfrak{r}_1} = (\beta, \vec{\zeta}) = (r \cdot \alpha_{\mathrm{key}}, \quad r \cdot M_1^\top \cdot \vec{\gamma}_{\mathrm{key}} + \langle \delta_{\mathrm{key}}, 0, \ldots, 0 \rangle^\top) \in \mathcal{G}_{\mathrm{key}}^{d+2}$$

Since K is an affine matrix, $K^\top$ will map vectors of the form $\langle \delta_{\mathrm{key}}, 0, \ldots, 0 \rangle^\top$ to other such vectors, so that

$$K^\top \cdot (r \cdot M_1^\top \cdot \vec{\gamma}_{\mathrm{key}} + \langle \delta_{\mathrm{key}}, 0, \ldots, 0 \rangle^\top) = (r \cdot M_2^\top \cdot \vec{\gamma}_{\mathrm{key}} + \langle \delta_{\mathrm{key}}, 0, \ldots, 0 \rangle^\top)$$

and $(\beta, K^\top \cdot \vec{\zeta})$ is a valid secret key for $\mathfrak{r}_2$. However, it is not a random secret key for $\mathfrak{r}_2$ because it re-uses the same r. To randomize r, DELEGATE chooses $r' \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and outputs

$$\mathrm{SK}_{\mathfrak{r}_2} \leftarrow (\beta + r' \cdot \alpha_{\mathrm{key}}, \; K^\top \cdot \vec{\gamma} + r' \cdot M_2^\top \cdot \vec{\gamma}_{\mathrm{key}})$$

Encrypt To implement ENCRYPT($\mathrm{PP}, \mathfrak{p}, m$), let $\mathfrak{p} = \mathrm{affspan}(M)$. In the case of spatial encryption, $\mathfrak{p}$ is a single point, so that M is an $(n+1) \times 1$ matrix; for doubly-spatial encryption it is an $(n+1) \times (d+1)$ matrix, where $d \in [0, n]$ is the dimension of $\mathfrak{p}$. ENCRYPT chooses a random $s \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and computes

$$\begin{aligned} \mathrm{HDR} &\leftarrow (s \cdot \alpha_{\mathrm{ct}}, \; s \cdot M^\top \cdot \vec{\gamma}_{\mathrm{ct}}) \in \mathcal{G}_{\mathrm{ct}}^{d+2} \\ \mathrm{MKEY} &\leftarrow s \cdot \tau = s \cdot \delta_{\mathrm{key}} \otimes \alpha_{\mathrm{ct}} \end{aligned}$$

It outputs HDR, and then encrypts the message using MKEY. For example, it may compute $m + \mathrm{MKEY}$, or it may hash MKEY into a secret key for some symmetric cipher and then use that cipher to encrypt m. In other words, encryption acts as a *key encapsulation mechanism*, or KEM. Note that the header contains $d+2$ group elements, so that in the case of spatial encryption its size is 2 group elements irrespective of n.

Decrypt The essence of DECRYPT$(\mathrm{PP}, \mathfrak{r}, \mathrm{SK}_{\mathfrak{r}}, \mathfrak{p}, c)$ is to recover MKEY. To do this, it finds an intersection point $\vec{x} \in \mathfrak{p} \cap \mathfrak{r} \subseteq \mathrm{Aff}\left(\mathbb{Z}_p^n\right)$ and runs DELEGATE$(\mathrm{PP}, \mathfrak{r}, \mathrm{SK}_{\mathfrak{r}}, \{\vec{x}\})$ to find a secret key for $\mathfrak{p}$, which is of the form

$$(r \cdot \alpha_{\mathrm{key}},\ r \cdot \vec{x}^{\top} \cdot \vec{\gamma}_{\mathrm{key}} + \delta_{\mathrm{key}})$$

Likewise, let $\mathfrak{p} = \mathrm{affspan}(M)$ for some matrix M; DECRYPT finds the vector $\vec{y}$ for which $\vec{x} = M \cdot \vec{y}$. The ciphertext's HDR is of the form

$$(\beta,\ \vec{\zeta}) = (s \cdot \alpha_{\mathrm{ct}},\ s \cdot M^{\top} \cdot \vec{\gamma}_{\mathrm{ct}})$$

so that

$$(\beta,\ \vec{y}^{\top}\vec{\zeta}) = (s \cdot \alpha_{\mathrm{ct}},\ s \cdot \vec{x}^{\top} \cdot \vec{\gamma}_{\mathrm{ct}})$$

Note that this amounts to a version of DELEGATE for ciphertexts, without the re-randomization. From these two values, DECRYPT can compute

$$\begin{aligned}(r \cdot \vec{x}^{\top}\vec{\gamma}_{\mathrm{key}} + \delta_{\mathrm{key}}) \otimes (s \cdot \alpha_{\mathrm{ct}}) - (r \cdot \alpha_{\mathrm{key}}) \otimes (s \cdot \vec{x}^{\top}\vec{\gamma}_{\mathrm{ct}}) &= \delta_{\mathrm{key}} \otimes (s \cdot \alpha_{\mathrm{key}}) \\ &= \mathrm{MKEY}\end{aligned}$$

by bilinearity of $\otimes$, and use MKEY to decrypt the message.

Now for doubly-spatial encryption, this algorithm does not quite match our definition for DECRYPT, because its output is not independent of the role $\mathfrak{r}$. It is independent of the secret key $\mathrm{SK}_{\mathfrak{r}}$ because of the use of DELEGATE, but the intersection point $\vec{x}$ is not independent of $\mathfrak{r}$. To fix this problem, we need to check that the purported $s \cdot M^{\top} \cdot \gamma_{\mathrm{ct}}$ is actually a multiple of $M^{\top} \cdot \gamma_{\mathrm{ct}}$. With high probability it suffices to check that $(s \cdot \alpha_{\mathrm{ct}}, s \cdot \vec{y} \cdot M^{\top} \cdot \vec{\gamma}_{\mathrm{ct}})$ is a multiple of $(\alpha_{\mathrm{ct}}, \vec{y} \cdot M^{\top} \cdot \vec{\gamma}_{\mathrm{ct}})$ for a single random $\vec{y} \in \mathbb{Z}_p^{d+1}$, which can be done using two pairing operations. Alternatively, we could force the sender to include a proof of knowledge for s, but this would increase ciphertext size.

5.3 Extended versions

Spatial encryption In the extended version of spatial encryption, we allow a policy to be a "positive" affine subspace $W = \text{affspan}(M) \subseteq \mathbb{Z}_p^n$ of dimension d, so that only a role containing W (rather than intersecting it) can decrypt the message. To do this, ENCRYPT chooses a random vector $\vec{u} \xleftarrow{\text{R}} \text{Aff}\left(\mathbb{Z}_p^d\right)$ in addition to s, and outputs the header as

$$\text{HDR} \leftarrow (s \cdot \vec{u} \cdot \alpha_{\text{ct}},\ s \cdot \vec{u}^\top \cdot M^\top \cdot \vec{\gamma}_{\text{ct}}) \in \mathcal{G}_{\text{ct}}^{d+2}$$

with the same $\text{MKEY} = s \cdot \tau$.

To decrypt this, DECRYPT runs DELEGATE$(\text{PP}, \mathfrak{r}, \text{SK}_{\mathfrak{r}}, \mathfrak{p})$ to obtain a secret key for the same subspace W; this will be of the form

$$(r \cdot \alpha_{\text{key}},\quad r \cdot M^\top \cdot \vec{\gamma}_{\text{key}} + \langle \delta_{\text{key}}, 0, \ldots, 0 \rangle^\top)$$

It then computes

$$\begin{aligned}
& (r \cdot M^\top \cdot \vec{\gamma}_{\text{key}} + \langle \delta_{\text{key}}, 0, \ldots, 0 \rangle^\top)^\top \otimes (s \cdot \vec{u} \cdot \alpha_{\text{ct}}) - (r \cdot \alpha_{\text{key}}) \otimes (s \cdot \vec{u}^\top \cdot M^\top \cdot \vec{\gamma}_{\text{ct}}) \\
= \quad & rs \cdot (\vec{\gamma}_{\text{key}}^\top \cdot M \cdot \vec{u}) \otimes \alpha_{\text{ct}} + \delta_{\text{key}} \otimes (s \cdot \alpha_{\text{ct}}) - rs \cdot \alpha_{\text{key}} \otimes (\vec{u}^\top \cdot M^\top \cdot \vec{\gamma}_{\text{ct}}) \\
= \quad & s \cdot \delta_{\text{key}} \otimes \alpha_{\text{ct}} \\
= \quad & \text{MKEY}
\end{aligned}$$

which completes the construction using $d + 2$ pairing operations.

We will note that (CPA, non-anonymous) security of this system follows from that of spatial encryption. To see this, first note that $\vec{u}$ can be re-randomized. In the unlikely but testable case that $s = 0$, this does nothing. Otherwise, choose $\vec{w} \xleftarrow{\text{R}} \{0\} \times \mathbb{Z}_p^d$. Add $\vec{w} \cdot \alpha_{\text{ct}}$ to the first component, and $\vec{w}^\top \cdot M^\top \cdot \vec{\gamma}_{\text{ct}}$ to the second; this effectively adds $\vec{w}/s$ to $\vec{u}$.

Now, consider what happens when the challenger is asked to produce a challenge ciphertext for some such $W = \text{affspan}(M)$. It can choose a random $\vec{u}$ itself, produce a challenge ciphertext for $M \cdot \vec{u}$, and then fill in $s \cdot \vec{u} \cdot \alpha_{\text{ct}}$ from $s \cdot \alpha_{\text{ct}}$ because it knows $\vec{u}$. Re-randomizing $\vec{u}$ then obliterates any trace of what $\vec{u}$ was chosen. Since the

adversary's delegation queries are prohibited from containing W, they do not contain $M \cdot \vec{u}$ (which is a random point in W) except with negligible probability ($\leq 1/p$ per query). Thus the adversary's attack against extended spatial encryption would work against spatial encryption proper.

Doubly-spatial encryption The case of extended doubly-spatial encryption is similar, except a role may now be a "negative" subspace $W = \text{affspan}(M)$, and can only decrypt policies on (negative) subspaces that contain W. The form of a secret key is

$$(\vec{r} \cdot \alpha_{\text{key}}, \quad \vec{r}^{\top} \cdot M^{\top} \cdot \vec{\gamma}_{\text{key}} + \delta_{\text{key}}) \in \mathcal{G}_{\text{key}}^{d+2}$$

Decryption works as in extended spatial encryption, again using $d+2$ pairings, so it remains to show how to delegate from $\mathfrak{r}_1$ to $\mathfrak{r}_2$ where $\mathfrak{r}_2$ is negative. If $\mathfrak{r}_1$ is positive, then we can delegate through a random point in $\mathfrak{r}_1 \cap \mathfrak{r}_2$, so it remains to solve the case when $\mathfrak{r}_1$ is negative.

Let $\mathfrak{r}_1 = \text{affspan}(M_1)$ and $\mathfrak{r}_2 = \text{affspan}(M_2)$; as before, we must have $M_1 = M_2 K$ for some affine matrix K. We are given

$$(\vec{r} \cdot \alpha_{\text{key}}, \quad \vec{r}^{\top} \cdot M_1^{\top} \cdot \vec{\gamma}_{\text{key}} + \delta_{\text{key}}) \in \mathcal{G}_{\text{key}}^{d+2}$$

Left-multiplying the first component by K gives

$$(K \cdot \vec{r} \cdot \alpha_{\text{key}}, \quad \vec{r}^{\top} \cdot M_1^{\top} \cdot \vec{\gamma}_{\text{key}} + \delta_{\text{key}}) = ((K \cdot \vec{r}) \cdot \alpha_{\text{key}}, \quad (K \cdot \vec{r})^{\top} \cdot M_2^{\top} \cdot \vec{\gamma}_{\text{key}} + \delta_{\text{key}})$$

This is a valid, but not uniformly random, key for $\mathfrak{r}_2$: its $\vec{r}$ value is $K \cdot \vec{r}$. Re-randomization works as in spatial encryption: DELEGATE chooses a random $\vec{r}'$, and adds $\vec{r}' \cdot \alpha_{\text{key}}$ to the first element and $\vec{r}'^{\top} \cdot M_2^{\top} \cdot \vec{\gamma}_{\text{key}} + \delta_{\text{key}}$ to the second.

Once again, the security of this system follows from that of doubly-spatial encryption. When asked to create a key for a negative subspace W, the challenger creates a key for a random point in that subspace, then delegates to create the final key. The challenge ciphertext cannot contain W, so it contains the chosen point with probability at most $1/p$.

Chapter 6

Proofs of Security

We offer two proofs of security of spatial encryption, when instantiated in groups with different properties. The first proves only selective security, and uses a complex (but standard) assumption. However, it is relatively easy to follow, and the groups which it uses are simple and fast. Furthermore, doubly-spatial encryption is shown to be selectively secure assuming that spatial encryption is selectively secure.[1]

The second proof offers adaptive security. Our assumptions in this proof are in some ways less complex than those in the first proof (in particular, they involve only a constant number of terms), but they are less natural assumptions. Furthermore, the second proof only works for spatial encryption; doubly-spatial encryption using the same groups may still only be selectively secure.

Both proofs will show CPA, non-anonymous security. However, we can recover CCA_2 security via Theorem 1.

6.1 Master Theorem of Generic Groups

We will use several assumptions, some of them novel, as the basis for our security proofs. We would like to argue that the problems we call "hard" in these assumptions

[1] It follows that doubly-spatial encryption is adaptively secure in the generic group model, simply because in that model the adversary learns nothing from the keys she is given. However, we do not take much stock in the generic group model, so we do not consider this worth proving formally.

are indeed difficult, and for that we will appeal to the generic group model. We sketch a "master theorem" of generic group security from Boneh and Boyen [6].

Suppose that an adversary is given a vector $\vec{\gamma}$ of k elements from a cyclic group $\mathcal{G}$ of prime order p, generated by some element α (which we will take to be included in $\vec{\gamma}$), and uses group operations to generate q more. These $k+q$ elements are all linear combinations of the original k elements with known coefficients. By assumption, the only information that the attacker uses about these $k+q$ elements is which ones are equal, i.e. which ones have zero difference. Thus the entire attack algorithm may be treated as $(k+q)(k+q-1)/2$ tests, each choosing a vector $\vec{v}$ of k elements from $\mathbb{Z}_p$ and asking whether $\vec{v}^\top\vec{\gamma} = 0$.

Suppose that $\vec{\gamma}$ is generated by choosing random elements $r_1, \ldots, r_n \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, then evaluating $P_i(r_1, \ldots, r_n) \cdot \alpha$ for each $i \in [1, k]$, where P_i is a fixed polynomial over $\mathbb{Z}_p$; together call these polynomials $\vec{P}$ and call the resulting distribution $\mathcal{D}_{\vec{P}}$. Then $\vec{v}^\top\vec{\gamma} = (\vec{v}^\top\vec{P})(r_1, \ldots, r_n)$. Now if $\vec{v}^\top\vec{P} = 0$ then this is sure to be zero (and the adversary learns nothing). Otherwise, it is easily seen to be zero with probability at most d/p, where d is the maximum total degree of the polynomials P_i. Thus the group elements will be unequal except with probability at most $d(k+q)(k+q-1)/2p = dq^2/2p + O(dkq/p)$. Here the first term dominates, so we may ignore the second.

Suppose that the adversary's goal is to distinguish $\mathcal{D}_{\vec{P}}$ from some other distribution $\mathcal{D}_{\vec{Q}}$. This will be easy if some $\vec{v}$ exists with $\vec{v}^\top\vec{P} = 0$ but $\vec{v}^\top\vec{Q} \neq 0$ or vice versa. Given $\vec{P}$ and $\vec{Q}$, the existence of such a $\vec{v}$ can easily be checked by hand or by computer. If no such $\vec{v}$ exists, then all non-tautological tests will come up unequal (and thus the adversary cannot distinguish the two distributions) except with probability about $dq^2/2p$, where d is the maximum total degree of all the P_i and Q_i. Therefore to distinguish the distributions with advantage ϵ requires on the order of $\sqrt{2\epsilon p/d}$ group operations.

A similar theorem holds in the case of groups with a bilinear pairing, except that the adversary's queries have the form $\vec{P}_{\mathrm{key}}^\top M \vec{P}_{\mathrm{ct}} + \vec{v}^\top \vec{P}_T$ for an arbitrary matrix M and vector $\vec{v}$, and the maximum total degrees of $\vec{P}_{\mathrm{ct}}$ and $\vec{P}_{\mathrm{key}}$ sum. Together, these theorems comprise the *master theorem* of the generic group model.

6.2 Selective Security

Assumption Our proof of selective security is based on that of the Boneh-Boyen-Goh HIBE [7], and uses an asymmetric variant of the $(n+1)$-*bilinear Diffie-Hellman exponent* (BDDHE) assumption from that work. In this assumption, we state that it is difficult to verify a computation on $x^{n+1} \cdot \alpha$ given other powers of x times α. More formally, we generate two distributions as follows:

$$\begin{aligned}
(p;\ \mathcal{G}_{\text{key}}, \mathcal{G}_{\text{ct}}, \mathcal{G}_T;\ \otimes;\ \alpha_{\text{ct}}, \alpha_{\text{key}}) &\leftarrow \textsc{GroupGen}(\lambda) \\
\text{GP} &\leftarrow (p;\ \mathcal{G}_{\text{key}}, \mathcal{G}_{\text{ct}}, \mathcal{G}_T;\ \otimes) \\
x, y, z &\xleftarrow{\text{R}} \mathbb{Z}_p^* \times \mathbb{Z}_p \times \mathbb{Z}_p \\
\vec{\chi}_{\text{ct}} &\leftarrow (1, x, x^2, \ldots, x^n) \cdot \alpha_{\text{ct}} \\
\vec{\chi}_{\text{key}} &\leftarrow (1, x, x^2, \ldots, x^n) \cdot \alpha_{\text{key}} \\
\vec{\eta}_{\text{key}} &\leftarrow (x^{n+2}, \ldots, x^{2n+2}) \cdot \alpha_{\text{key}} \\
\mathcal{D}_0 &\leftarrow (\text{GP};\ \vec{\chi}_{\text{ct}}, \vec{\chi}_{\text{key}}, \vec{\eta}_{\text{key}};\ y \cdot \alpha_{\text{ct}};\ x^{n+1} y \cdot \alpha_{\text{key}} \otimes \alpha_{\text{ct}}) \\
\mathcal{D}_1 &\leftarrow (\text{GP};\ \vec{\chi}_{\text{ct}}, \vec{\chi}_{\text{key}}, \vec{\eta}_{\text{key}};\ y \cdot \alpha_{\text{ct}};\ z \cdot \alpha_{\text{key}} \otimes \alpha_{\text{ct}})
\end{aligned}$$

Note that the last two elements of $\mathcal{D}_1$ are uniformly random. These distributions differ from those in the original $(n+1)$-BDDHE assumption in that they allow $\mathcal{G}_{\text{ct}}$ and $\mathcal{G}_{\text{key}}$ to be different. If $\mathcal{G}_{\text{ct}} = \mathcal{G}_{\text{key}}$ and $\alpha_{\text{ct}} = \alpha_{\text{key}}$, then $\vec{\chi}_{\text{ct}} = \vec{\chi}_{\text{key}}$ and we recover the original $(n+1)$-BDDHE assumption. Thus spatial encryption is secure under this assumption as well. Furthermore, we note that we have chosen a BDDHE-type assumption for its relative simplicity; a weaker $(n+1)$-extended BDDH assumption [25] would have worked as well.

Given an algorithm $\mathcal{A}$ which accepts these distributions and returns 0 or 1, we define its $(n+1)$-optionally-asymmetric-BDDHE-advantage as

$$\text{Adv}(\mathcal{A} \leftrightarrow (n+1)\text{-OA-BDDHE}) = |\Pr[\mathcal{A}(x) = 1 : x \leftarrow \mathcal{D}_1] - \Pr[\mathcal{A}(x) = 1 : x \leftarrow \mathcal{D}_0]|$$

We assume that for reasonable sizes of n (say, 10,000) and λ (say, 128), no feasible algorithm $\mathcal{A}$ has a non-negligible $(n+1)$-OA-BDDHE advantage.

The master theorem shows the $(n+1)$-OA-BDDHE assumption to be true in the generic group model, requiring on the order of $\sqrt{2\epsilon p/3n}$ time to break with advantage ϵ. Alternatively, the $(n+1)$-extended BDDH assumption can be used, which requires on the order of $\sqrt{\epsilon p/2}$ time to break with advantage ϵ.

We are now ready to prove the security of spatial encryption.

Theorem 3. *Let $\mathcal{A}$ be a (selective, CPA, non-anonymous) adversary against n-dimensional spatial encryption. Then there is an adversary $\mathcal{B}_1$ against the underlying symmetric encryption system $\mathbf{E}$ and an $(n+1)$-OA-BDDHE adversary $\mathcal{B}_2$, both running in about the same time as $\mathcal{A}$, such that*

$$\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{Sp}) \leq \mathrm{Adv}(\mathcal{B}_1 \leftrightarrow \mathbf{E}) + 2 \cdot \mathrm{Adv}(\mathcal{B}_2 \leftrightarrow (n+1)\text{−OA−BDDHE})$$

Proof. We will prove security in two stages. First, we will show that substituting arbitrary parameters for random ones can only hurt security. Second, we will reduce spatial encryption to the $(n+1)$-OA-BDDHE assumption by setting up the system from that assumption.

Rigged parameters We will use variants of $\vec{\chi}_{\text{key}}$ and $\vec{\chi}_{\text{ct}}$ as the $\vec{\gamma}$ components of the system's public parameters, even though they are not uniformly random. Here we show that an algorithm which breaks spatial encryption with random parameters can be adapted to break one with rigged parameters. To do this, choose $\vec{r} \xleftarrow{\text{R}} \mathbb{Z}_p^{n+1}$, and set

$$\vec{\gamma}\,'_{\text{key}} \leftarrow \vec{\gamma}_{\text{key}} + \vec{r} \cdot \alpha_{\text{key}} \quad \text{and} \quad \vec{\gamma}\,'_{\text{ct}} \leftarrow \vec{\gamma}_{\text{ct}} + \vec{r} \cdot \alpha_{\text{ct}}$$

so that $\vec{\gamma}\,'_{\text{key}}$ and $\vec{\gamma}\,'_{\text{ct}}$ are each uniformly random (but are still proportional to each other). Run the adversary with $\vec{\gamma}\,'_{\text{key}}$ and $\vec{\gamma}\,'_{\text{ct}}$ in the public parameters instead of $\vec{\gamma}_{\text{key}}$ and $\vec{\gamma}_{\text{ct}}$. When the adversary makes a delegation query, note that $r \cdot \alpha_{\text{key}}$ is returned along with $r \cdot M^\top \cdot \vec{\gamma}_{\text{key}} + \langle \delta_{\text{key}}, 0, \ldots, 0 \rangle^\top$ for some matrix M which is known to the simulator. Thus adding $M^\top \cdot \vec{r} \cdot \alpha_{\text{key}}$ makes the result consistent with $\vec{\gamma}\,'_{\text{key}}$. Likewise, a challenge ciphertext can be made consistent with $\vec{\gamma}\,'_{\text{ct}}$.

A similar argument shows that δ_{key} can also be rigged. Choose a random $t \in \mathbb{Z}_p^*$

and set $\delta'_{\text{key}} \leftarrow t \cdot \delta_{\text{key}}$, so that δ'_{key} is uniformly random in $\mathcal{G}_{\text{key}} \backslash \{0\}$. Adjust the public parameters with $\tau' \leftarrow t \cdot \tau$. Adjust every delegation query by multiplying by t, and adjust the challenge ciphertext by dividing the header by t.

We are now ready to prove security of spatial encryption by a sequence of games.

Game 0 We call the real security game Game 0.

Game 1 This game is exactly the same as Game 0, but the challenger computes its responses differently. It begins with rigged parameters sampled from $\mathcal{D}_0$, and then re-randomizes them as described above to produce a game which is distributed identically to Game 0. SETUP samples

$$(\text{GP};\ \vec{\chi}_{\text{ct}}, \vec{\chi}_{\text{key}}, \vec{\eta}_{\text{key}};\ y \cdot \alpha_{\text{ct}};\ x^{n+1} y \cdot \alpha_{\text{key}} \otimes \alpha_{\text{ct}}) \leftarrow \mathcal{D}_0$$

It sends GP to the adversary, but waits to compute PP until it learns the challenge policy $\mathfrak{p} \in \text{Aff}\left(\mathbb{Z}_p^n\right)$. It then chooses

$$\vec{\gamma}_{\text{ct}} = \vec{\chi}_{\text{ct}} - \left\langle \mathfrak{p}^\top \vec{\chi}_{\text{ct}}, 0, \ldots, 0 \right\rangle^\top$$

so that $\mathfrak{p}^\top \vec{\gamma}_{\text{ct}} = 0$, and likewise

$$\vec{\gamma}_{\text{key}} = \vec{\chi}_{\text{key}} - \left\langle \mathfrak{p}^\top \vec{\chi}_{\text{key}}, 0, \ldots, 0 \right\rangle^\top$$

SETUP does not produce a master key. It instead conceptually sets $\delta_{\text{key}} = x^{n+1} \cdot \alpha_{\text{key}}$, which is a value that it does not know.

When the adversary asks a delegation query for a role

$$\mathfrak{r} = W = \text{affspan}(M) \ \text{ where } \ M = \begin{pmatrix} 1 & \\ \vec{u} & L \end{pmatrix}$$

the challenger will need to compute a response in some different way, since it does not have the master secret key. Let $\vec{v} \in \mathbb{Z}_p^n$ be a vector which is normal to L, but such that $\vec{v}^\top \mathfrak{p} \neq \vec{v}^\top \vec{u}$. Such a $\vec{v}$ is guaranteed to exist when $\mathfrak{p} \notin W$, and can be

computed by Gaussian elimination. Furthermore, by scaling $\vec{v}$, we can arrange that $\vec{v}^\top \mathfrak{p} - \vec{v}^\top \vec{u} = 1$.

Now, conceptually let

$$r \leftarrow \vec{v}^\top \cdot \left\langle x^n, x^{n-1}, \ldots, x \right\rangle^\top \in \mathbb{Z}_p$$

The challenger will use this r to construct the secret key, despite not knowing its value. Recall that the secret key has the form

$$\begin{aligned} & (r \cdot \alpha_{\text{key}}, \quad r \cdot M^\top \cdot \vec{\gamma}_{\text{key}} + \langle \delta_{\text{key}}, 0, \ldots, 0 \rangle^\top) \\ = \; & (r, \quad r \cdot (\vec{u} - \mathfrak{p})^\top \cdot \left\langle x, x^2, \ldots, x^n \right\rangle^\top + x^{n+1}, \quad r \cdot L^\top \cdot \left\langle x, x^2, \ldots, x^n \right\rangle^\top) \cdot \alpha_{\text{key}} \end{aligned}$$

We will now explain how to compute these terms. They are all α_{key} times polynomials in x of degree at most $2n$, so we can compute them if their x^{n+1} coefficients are 0. The first term, r, has degree only n. For the second and third terms, note that the x^{n+1} coefficient on $r \cdot \vec{z}^\top \cdot \langle x, x^2, \ldots, x^n \rangle^\top$ is precisely $\vec{z}^\top \cdot \vec{v}$. So in the second term, the coefficient is $-1 + 1 = 0$, and on the third term it is 0 because $\vec{v}$ is orthogonal to L. Thus the challenger can compute all these terms. To make r uniformly random, the challenger then rerandomizes the secret key as in DELEGATE. Finally, the challenger adjusts the secret key from the rigged parameters to the uniformly random ones.

To generate the challenge ciphertext, the challenger conceptually sets $s = y$, so that the header is

$$(y \cdot \alpha_{\text{ct}}, \quad y \cdot \mathfrak{p}^\top \cdot \vec{\gamma}_{\text{ct}}) \quad = \quad (y \cdot \alpha_{\text{ct}}, \quad 0)$$

since $\vec{\gamma}_{\text{ct}}$ was chosen to be orthogonal to $\mathfrak{p}$. The MKEY is then $x^{n+1} y \cdot \alpha_{\text{key}} \otimes \alpha_{\text{ct}}$. These values are available from the sample of $\mathcal{D}_0$. Once again, the challenger adjusts from the rigged parameters to the random ones, and returns the challenge ciphertext to the adversary.

Game 1 produces public parameters, secret keys and challenge ciphertexts according to the same distribution as Game 0, so the adversary's advantage in both cases is the same.

Game 2 Our final game is the same as Game 1, except that the challenger uses $\mathcal{D}_1$ instead $\mathcal{D}_0$. In this case, everything will look the same except that the computed MKEY will be uniformly random, so that the adversary's advantage in Game 2 is the same as her advantage against the underlying symmetric encryption system (which is 0 in the case of that MKEY is simply added to m, but may be nonzero otherwise). That is, there is an adversary $\mathcal{B}_1$ against the underlying encryption scheme $\mathbf{E}$ such that

$$\mathrm{Adv}(\mathcal{A} \leftrightarrow \text{Game 2}) = \mathrm{Adv}(\mathcal{B}_1 \leftrightarrow \mathbf{E})$$

However, distinguishing Game 1 and Game 2 breaks the $(n+1)$-OA-BDDHE problem. That is, there is an $(n+1)$-OA-BDDHE adversary $\mathcal{B}_2$ such that

$$\mathrm{Adv}(\mathcal{B}_2 \leftrightarrow (n+1)\text{-OA-BDDHE}) \geq \frac{1}{2}\left(\mathrm{Adv}(\mathcal{A} \leftrightarrow \text{Game 2}) - \mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{Sp})\right)$$

Combining these, we get

$$\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{Sp}) \leq \mathrm{Adv}(\mathcal{B}_1 \leftrightarrow \mathbf{E}) + 2 \cdot \mathrm{Adv}(\mathcal{B}_2 \leftrightarrow (n+1)\text{-OA-BDDHE})$$

as claimed. □

Doubly-spatial encryption It is even easier to show that the selective security of doubly spatial encryption and of spatial encryption are the same.

Theorem 4. *Let $\mathcal{A}$ be a (selective, CPA, non-anonymous) adversary against n-dimensional doubly-spatial encryption. Then there is another (selective, CPA, non-anonymous) adversary $\mathcal{B}$ against n-dimensional spatial encryption, running in about the same time as $\mathcal{A}$, such that*

$$\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{DblSp}) = \mathrm{Adv}(\mathcal{B} \leftrightarrow \mathbf{Sp})$$

Proof. Doubly-spatial encryption is the same as spatial encryption, except that the chosen policy may be a d-dimensional subspace $\mathfrak{p} = \mathrm{affspan}(M)$ instead of a single point. At a high level, the simulator will project $\mathfrak{p}$ down to a single point, reducing

to spatial encryption in $n-d$ dimensions. More formally, let $K \in \mathbb{Z}_p^{(n-d+1)\times(n+1)}$ be a surjective affine matrix with $KM = \langle 1, 0, \ldots, 0\rangle^\top \in \text{Aff}\left(\mathbb{Z}_p^{n-d}\right)$, so that the chosen policy $\mathfrak{p}$ (and nothing else) maps to the affine origin.

The simulator sets up spatial encryption in $n-d$ dimensions, selecting the origin as the policy, and receives $\vec{\gamma}_{\text{ct}}$ and $\vec{\gamma}_{\text{key}}$ in the public parameters. It then rigs its own public parameters with $K^\top \cdot \vec{\gamma}_{\text{ct}}$ and $K^\top \cdot \vec{\gamma}_{\text{key}}$. Note that for any affine matrix $L \in \mathbb{Z}_p^{(n+1)\times\ell}$, we have $(KL)^\top \cdot \vec{\gamma}_{\text{key}} = L^\top \cdot K^\top \cdot \vec{\gamma}_{\text{key}}$, so that a secret key for affspan(L) in the "outer" system is the same as one for the its projection affspan(KL) in the "inner" one. So the simulator for the outer system left-multiplies all queries by K, then passes them to the inner system, adjusting the returned secret keys for rigged parameters as usual. It does the same for the challenge ciphertext. The queries will still be valid: in the outer system affspan(L) does not intersect $\mathfrak{p}$, so its image will not contain $\langle 1, 0, \ldots, 0\rangle^\top$.

This completes the simulation of doubly-spatial encryption using spatial encryption in $n-d$ dimensions. To complete the proof that the (selective, CPA, non-anonymous) security of n-dimensional doubly-spatial encryption is the same as that of n-dimensional spatial encryption, note that the inner $(n-d)$-dimensional spatial encryption system can be emulated by fixing the last d dimensions of an n-dimensional spatial encryption system to 0. □

Multiple masters Since the only part of setup that is secret is the choice of δ_{key}, we can consider the case in which the remaining parameters, GP, α_{ct}, α_{key}, $\vec{\gamma}_{\text{ct}}$ and $\vec{\gamma}_{\text{key}}$ are chosen by a trusted third party or, equivalently, as the output of a random-oracle hash. In this case, anyone can set up a private copy of the system by generating a random δ_{key} and publishing $\delta_{\text{key}} \otimes \alpha_{\text{ct}}$. Each of these systems will then be selectively secure, by the same argument.

Furthermore, suppose two systems are created in this way, with public parameters PP and PP′. Then given a ciphertext for a policy $\mathfrak{p}$, it will not be possible to tell which system created it without the ability to decrypt in either system. This is because only MKEY depends on δ_{key}, and it is indistinguishable from random without a key for a role $\mathfrak{r} \succeq \mathfrak{p}$.

Thresholds for the master secret Alternatively, a coalition of k authorities can generate k different shares of δ_{key}, such that any t of them can recover δ_{key} but $t-1$ cannot. For example, following Shamir [34] they can generate a random polynomial P of degree $t-1$ and set $\sigma_i \leftarrow P(i) \cdot \alpha_{\text{key}}$ for each $i \neq 0$. The master secret key is then $\delta_{\text{key}} \leftarrow P(0) \cdot \alpha_{\text{key}}$, which can be recovered from t shares by interpolation, but any $t-1$ shares give no information about δ_{key}.

Sharing commutes with delegation. That is, for any role $\mathfrak{r}$, each authority can generate a secret key for $\mathfrak{r}$ with σ_i replacing δ_{key}, and performing interpolation term-by-term on t these keys gives a secret key for $\mathfrak{r}$. This provides a convenient way for users to obtain secret keys from the authorities. The shares of these secret keys can even be verified if each authority publishes $\sigma_i \otimes \alpha_{\text{ct}}$.

Such a system is secure under *selective authority compromise*. That is, if the adversary compromises $t-1$ authorities before the game begins, she still has no advantage. To prove this, we can simulate the scenario as follows:

- The simulator instantiates spatial encryption as usual.
- The simulator chooses σ_i for each compromised authority i and gives it to the adversary. It uses interpolation on $\delta_{\text{key}} \otimes \alpha_{\text{ct}}$ and the $t-1$ compromised σ_i values to compute $\sigma_j \otimes \alpha_{\text{ct}}$ for each uncompromised authority j.
- When the adversary queries a compromised authority i for secret key share to a role $\mathfrak{r}$, the simulator generates it directly from σ_i. When she queries an uncompromised authority j, the simulator generates the real secret key for $\mathfrak{r}$, and a share for each compromised authority i, then interpolates these t keys to generate the secret key from authority j.

This simulation produces the same distribution of secret key shares as the real system. Similarly, if the adversary is fully adaptive, we wish to show that she cannot decrypt a message encrypted to $\mathfrak{p}$ unless she has queried roles $\mathfrak{r} \succeq \mathfrak{p}$ from at least t different authorities. We can do this with probability $1/\binom{k}{t-1}$ simply by guessing which $t-1$ authorities will be used, then applying the above simulation.

Finally, note that the sharing need not be done at the level of the master secret. Any key can be split into k shares, of which t are needed to use it, and those shares

can be further delegated. However, if this is done many times, then the argument above incurs an exponential loss of tightness unless the adversary chooses, at the time the shares are created, which ones she will compromise.

6.3 Adaptive Security

The proof given above is relatively simple and is applicable both to spatial and to doubly-spatial encryption. However, it only proves selective security. Here we will show that, when instantiated with appropriate groups, spatial encryption is adaptively secure under a few compact assumptions. We will demonstrate compact assumptions, Assumption 1 through Assumption 3, all of which are true in the generic group model. We will then have the following theorem:

Theorem 5. *For any (CPA, adaptive, non-anonymous) adversary $\mathcal{A}$ against spatial encryption, there are four adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_{\mathbf{E}}$ against Assumption 1 through Assumption 3 and the symmetric encryption scheme $\mathbf{E}$, respectively, such that*

$$\begin{aligned}\mathrm{Adv}(\mathcal{A} \leftrightarrow \mathbf{Sp}) \;\leq\; & 2 \cdot \mathrm{Adv}(\mathcal{B}_1 \leftrightarrow \text{Assumption 1}) + 2q \cdot \mathrm{Adv}(\mathcal{B}_2 \leftrightarrow \text{Assumption 2}) \\ & + 2 \cdot \mathrm{Adv}(\mathcal{B}_3 \leftrightarrow \text{Assumption 3}) + \mathrm{Adv}(\mathcal{B}_{\mathbf{E}} \leftrightarrow \mathbf{E})\end{aligned}$$

We will follow the "dual-system" technique presented in [26]. However, unlike that work, we will use prime-order assumptions.

Proof outline The dual-system encryption technique, pioneered by Brent Waters in [37], follows a completely different outline from the security proof given above. Unlike in a selective-security proof, the simulated challenger knows the master secret key and can answer any query. However, it subtly corrupts the challenge ciphertext and some of the secret keys it produces, making them *semi-functional*. Semi-functional keys can decrypt ordinary ciphertexts and vice-versa, but semi-functional keys cannot usually decrypt semi-functional ciphertexts.

There is an exception, however. Each semi-functional key or ciphertext is given a *tag*, which in our case is a random vector in $\mathbb{Z}_p^{n+1}$. Given a semi-functional ciphertext

for a policy $\mathfrak{p}$, and a semi-functional secret key for a role $\mathfrak{r} \succeq \mathfrak{p}$, if the tags on the key and ciphertext are the same then decryption will work as usual. However, if $\mathfrak{r} \not\succeq \mathfrak{p}$, then it is impossible to determine whether the tags match.

Group properties We require some additional properties from the groups $\mathcal{G}_{\text{key}}$ and $\mathcal{G}_{\text{ct}}$ for this proof. Instead of being cyclic, we require $\mathcal{G}_{\text{key}}$ to be generated by α_{key} and a new, "pernicious" element π_{key}, and likewise, $\mathcal{G}_{\text{ct}}$ should be generated by α_{ct} and π_{ct}. These pernicious elements must be orthogonal to the ordinary ones, meaning that

$$\alpha_{\text{key}} \otimes \pi_{\text{ct}} = \pi_{\text{key}} \otimes \alpha_{\text{ct}} = 0$$

even though

$$\alpha_{\text{key}} \otimes \alpha_{\text{ct}} \neq 0 \quad \text{and} \quad \pi_{\text{key}} \otimes \pi_{\text{ct}} \neq 0$$

The ordinary GroupGen does not output π_{key} and π_{ct}, and in fact they must be difficult to compute from α_{key} and α_{ct}. However, we assume that an enhanced algorithm, GroupGenSim, will output the same distribution as GroupGen with the addition of π_{key} and π_{ct}.

To achieve this, let $\mathcal{G}_{\text{key}}$ and $\mathcal{G}_{\text{ct}}$ be generated as follows. Choose cyclic groups $\mathcal{H}_{\text{key}}$ and $\mathcal{H}_{\text{ct}}$ of order p, generated by η_{key} and η_{ct}, respectively, supporting a bilinear map $\otimes : \mathcal{H}_{\text{key}} \times \mathcal{H}_{\text{ct}} \to \mathcal{G}_T$. $\mathcal{H}_{\text{key}}$ and $\mathcal{H}_{\text{ct}}$ may be the same group or different groups. We will set $\mathcal{G}_{\text{key}}$ and $\mathcal{G}_{\text{ct}}$ to be randomized subgroups of $\mathcal{H}_{\text{key}}^4$ and $\mathcal{H}_{\text{ct}}^4$, respectively, with their pairing to $\mathcal{G}_T$ being an "inner product" as described in Section 5.1. Choose matrices M_{key} and M_{ct} uniformly from $GL_4(\mathbb{Z}_p)$, the group of invertible 4×4 matrices over $\mathbb{Z}_p$, with the constraint that $M_{\text{key}}^\top \cdot M_{\text{ct}}$ is diagonal. Set

$$\begin{aligned}
\alpha_{\text{key}} &\leftarrow M_{\text{key}} \cdot \langle 1,0,0,0 \rangle^\top \cdot \eta_{\text{key}} \\
\pi_{\text{key}} &\leftarrow M_{\text{key}} \cdot \langle 0,1,0,0 \rangle^\top \cdot \eta_{\text{key}} \\
\beta_{\text{key}} &\leftarrow M_{\text{key}} \cdot \langle 0,0,1,0 \rangle^\top \cdot \eta_{\text{key}} \\
\alpha_{\text{ct}} &\leftarrow M_{\text{ct}} \cdot \langle 1,0,0,0 \rangle^\top \cdot \eta_{\text{ct}} \\
\pi_{\text{ct}} &\leftarrow M_{\text{ct}} \cdot \langle 0,1,0,0 \rangle^\top \cdot \eta_{\text{ct}} \\
\beta_{\text{ct}} &\leftarrow M_{\text{ct}} \cdot \langle 0,0,0,1 \rangle^\top \cdot \eta_{\text{ct}}
\end{aligned}$$

This gives the desired properties, with the additional property that the β elements are orthogonal to everything, i.e.

$$\beta_{\text{key}} \otimes \alpha_{\text{ct}} = \beta_{\text{key}} \otimes \pi_{\text{ct}} = \beta_{\text{key}} \otimes \beta_{\text{ct}} = \alpha_{\text{key}} \otimes \beta_{\text{ct}} = \pi_{\text{key}} \otimes \beta_{\text{ct}} = 0$$

Then set

$$\mathcal{G}_{\text{key}} \leftarrow \mathcal{H}^4_{\text{key}} \,/_{\text{R}}\, \text{span}(\beta_{\text{key}}) \quad \text{and} \quad \mathcal{G}_{\text{ct}} \leftarrow \mathcal{H}^4_{\text{ct}} \,/_{\text{R}}\, \text{span}(\beta_{\text{ct}})$$

Because the β terms are orthogonal to everything, the inner-product pairing is still well-defined. The randomizing factors β_{key} and β_{ct} are part of the descriptions of $\mathcal{G}_{\text{key}}$ and $\mathcal{G}_{\text{ct}}$ respectively, and so will be output by GroupGen and GroupGenSim.

Before proceeding, we will note that M_{key} and M_{ct} can be "rigged", meaning that if the system is secure with a particular choice of M_{key} and M_{ct}, then it is secure with a random choice of M_{key} and M_{ct}. To see this, choose a random $L \in GL_4(\mathbb{Z}_p)$ and a random invertible diagonal matrix D, and let

$$M'_{\text{key}} \leftarrow L \cdot M_{\text{key}} \quad \text{and} \quad M'_{\text{ct}} \leftarrow L^{-1} \cdot M_{\text{ct}} \cdot D$$

Then ${M'_{\text{key}}}^{\top} \cdot M'_{\text{ct}} = M_{\text{key}}^{\top} \cdot M_{\text{ct}} \cdot D$ is still diagonal (and invertible), and M'_{key} and M'_{ct} are uniform in $GL_4(\mathbb{Z}_p)$ subject to this property. To rig the system, we must adjust parameters, keys and ciphertexts which are already generated to use the new M'_{key} and M'_{ct}. For keys, we will multiply each parameter by L, and for ciphertexts by $L^{-1} \cdot D_{11}$, where D_{11} is the upper-left corner of D. This correctly maps the old key parameters to the new ones, and the old α_{ct} to the new one. It maps β_{ct} and π_{ct} to multiples of the new ones; however, these terms are always scaled randomly before being used, so the resulting distributions will be the same.

Semi-functional keys and ciphertexts To construct semi-functional keys and ciphertexts, we add multiples of π_{key} and π_{ct} to ordinary keys and ciphertexts. A semi-functional key for a role $\mathfrak{r} = \text{affspan}(K)$ with tag $(t_0, \vec{t}) \in \mathbb{Z}_p \times \mathbb{Z}_p^{n+1}$ has the

form

$$(r \cdot \alpha_{\text{key}} + r' \cdot t_0 \cdot \pi_{\text{key}}, \quad K^\top \cdot (r \cdot \vec{\gamma}_{\text{key}} + r' \cdot \vec{t} \cdot \pi_{\text{key}}) + \langle \delta_{\text{key}}, 0, \ldots, 0 \rangle^\top) \in \mathcal{G}_{\text{key}}^{d+2}$$

where $r, r' \xleftarrow{\text{R}} \mathbb{Z}_p$. Likewise, a semi-functional ciphertext for a policy $\mathfrak{p} = \vec{v} \in \text{Aff}\left(\mathbb{Z}_p^n\right)$ with tag $(t_0, \vec{t})$ has the form

$$\begin{aligned} \text{HDR} &\leftarrow (s \cdot \alpha_{\text{ct}} + s' \cdot t_0 \cdot \pi_{\text{ct}},\ \vec{v}^\top \cdot (s \cdot \vec{\gamma}_{\text{ct}} + s' \cdot \vec{t} \cdot \pi_{\text{ct}})) \in \mathcal{G}_{\text{ct}}^{d+2} \\ \text{MKEY} &\leftarrow s \cdot \tau = s \cdot \delta_{\text{key}} \otimes \alpha_{\text{ct}} \end{aligned}$$

When using an ordinary key to decrypt a semi-functional ciphertext, the ciphertext's π_{ct} terms are irrelevant because the key only has α_{key} components, and $\alpha_{\text{key}} \otimes \pi_{\text{ct}} = 0$; thus decryption will work as normal. Similarly, a semi-functional key can decrypt an ordinary ciphertext. Finally, if the tags on the key and the ciphertext are the same, then decryption will work as normal.

Security assumptions Our assumptions all take the form of distinguishing certain group elements from random, given other group elements. Let $\mathcal{A}$ be an algorithm which can take as input either of two distributions. Its advantage in distinguishing those distributions is

$$\text{Adv}(\mathcal{A} \leftrightarrow (\mathcal{D}_0, \mathcal{D}_1) := |\Pr(\mathcal{A}(x) = 1 : x \xleftarrow{\text{R}} \mathcal{D}_0) - \Pr(\mathcal{A}(x) = 1 : x \xleftarrow{\text{R}} \mathcal{D}_1)|$$

Our assumptions are symmetric, meaning that our assumptions work even when $\mathcal{H}_{\text{ct}} = \mathcal{H}_{\text{key}} = \mathcal{H}$ with generator η; when $\mathcal{H}_{\text{ct}}$ and $\mathcal{H}_{\text{key}}$ are different, we are given the same terms multiplied by η_{ct} and η_{key}. We give two different versions of the first two assumptions: one with multivariate polynomials but of degree 1 in each variable, and one with fewer variables and fewer terms but of higher degree. This assumption is formed by substitutions in the original assumption.

Assumption 1 Let $r, s, t, u, v \xleftarrow{R} \mathbb{Z}_p$. Then given

$$(1, r, s, rs, t, u, su, rsu, ru + v) \cdot \eta$$

it is hard to distinguish $tv \cdot \eta$ from random. Alternatively, given

$$(1, r, r^2, r^3 + rv, t) \cdot \eta$$

it is hard to distinguish $tv \cdot \eta$ from random.

Assumption 2 Let $r, s, t, u, v \xleftarrow{R} \mathbb{Z}_p$. Then given

$$(1, r, s, rs, u, ru + t, su, rsu, st, tv) \cdot \eta$$

it is hard to distinguish $(ru + v) \cdot \eta$ from random. Alternatively, given

$$(1, r, r^2, r^3 + rt, t, tv) \cdot \eta$$

it is hard to distinguish $(r^3 + v) \cdot \eta$ from random.

Assumption 3 Let $r, s, t, u \xleftarrow{R} \mathbb{Z}_p$. Then given

$$(1, r, s, t, rs, ru, u + st) \cdot \eta$$

it is hard to distinguish $tu \cdot \eta \otimes \eta$ from random.

Using the master theorems and a simple computer program, we checked that these assumptions are true in the generic group model, each requiring on the order of $\sqrt{\epsilon p/3}$ group operations to break with advantage ϵ.

Setup using security assumptions To apply our three security assumptions, we use them to prove three lemmas. In each case, we emulate part of the SETUP algorithm, producing GP, KBP and (a partial version of) EP. But we also produce other, tagged versions of the parameters, and the lemmas show that it is difficult to

determine how the tagged parameters were generated.

The first lemma shows that under Assumption 1, it is difficult to distinguish a scaled copy of the encryption parameters from a scaled and tagged copy.

Lemma. *Let $\mathcal{D}_{10}$ and $\mathcal{D}_{11}$ be defined as follows:*

$$\begin{aligned}
(p;\ \mathcal{G}_{\text{key}}, \mathcal{G}_{\text{ct}}, \mathcal{G}_T;\ \otimes;\ \alpha_{\text{ct}}, \alpha_{\text{key}};\ \pi_{\text{ct}}, \pi_{\text{key}}) &\leftarrow \textsc{GroupGenSim}(\lambda) \\
\vec{r} &\xleftarrow{\text{R}} \mathbb{Z}_p^{n+1} \\
(t_0,\ \vec{t}) &\xleftarrow{\text{R}} \mathbb{Z}_p^* \times \mathbb{Z}_p^{n+1} \\
u &\xleftarrow{\text{R}} \mathbb{Z}_p \\
\vec{\gamma}_{\text{key}} &\leftarrow \vec{r} \cdot \alpha_{\text{key}} \\
\vec{\gamma}_{\text{ct}} &\leftarrow \vec{r} \cdot \alpha_{\text{ct}}
\end{aligned}$$

$$\begin{aligned}
\text{GP} &\leftarrow (p;\ \mathcal{G}_{\text{key}}, \mathcal{G}_{\text{ct}}, \mathcal{G}_T;\ \otimes) \\
C &\leftarrow (\text{GP};\quad \alpha_{\text{key}}, \vec{\gamma}_{\text{key}};\quad \alpha_{\text{ct}}, \vec{\gamma}_{\text{ct}}) \\
\mathcal{D}_{10} &\leftarrow (C,\quad u \cdot \alpha_{\text{ct}},\ u \cdot \vec{\gamma}_{\text{ct}}) \\
\mathcal{D}_{11} &\leftarrow (C,\quad u \cdot \alpha_{\text{ct}} + t_0 \cdot \pi_{\text{ct}},\ u \cdot \vec{\gamma}_{\text{ct}} + \vec{t} \cdot \pi_{\text{ct}})
\end{aligned}$$

Then if an algorithm $\mathcal{A}$ distinguishes $\mathcal{D}_{10}$ from $\mathcal{D}_{11}$ with advantage ϵ, then another algorithm $\mathcal{B}$ takes about the same time as $\mathcal{A}$ to break Assumption 1 with advantage ϵ.

The second lemma shows that under Assumption 2, given π_{key} and a copy of the encryption parameters tagged with $(t_0, \vec{t})$, it is difficult to distinguish a scaled copy of KBP from a copy which is tagged with $(t_0, \vec{t})$.

Lemma. *Let $\mathcal{D}_{20}$ and $\mathcal{D}_{21}$ be defined as follows:*

$$
\begin{aligned}
(p;\ \mathcal{G}_{\text{key}}, \mathcal{G}_{\text{ct}}, \mathcal{G}_T;\ \otimes;\ \alpha_{\text{ct}}, \alpha_{\text{key}};\ \pi_{\text{ct}}, \pi_{\text{key}}) &\leftarrow \textsc{GroupGenSim}(\lambda) \\
\vec{r} &\xleftarrow{\text{R}} \mathbb{Z}_p^{n+1} \\
(t_0,\ \vec{t}) &\xleftarrow{\text{R}} \mathbb{Z}_p^* \times \mathbb{Z}_p^{n+1} \\
u_1, u_2, u_3 &\xleftarrow{\text{R}} \mathbb{Z}_p \\
\vec{\gamma}_{\text{key}} &\leftarrow \vec{r} \cdot \alpha_{\text{key}} \\
\vec{\gamma}_{\text{ct}} &\leftarrow \vec{r} \cdot \alpha_{\text{ct}}
\end{aligned}
$$

$$
\begin{aligned}
\text{GP} &\leftarrow (p;\ \mathcal{G}_{\text{key}}, \mathcal{G}_{\text{ct}}, \mathcal{G}_T;\ \otimes) \\
C &\leftarrow (\text{GP};\quad \alpha_{\text{key}}, \vec{\gamma}_{\text{key}};\quad \alpha_{\text{ct}}, \vec{\gamma}_{\text{ct}};\quad u_1 \cdot \alpha_{\text{ct}}, u_1 \cdot \vec{\gamma}_{\text{ct}} + \vec{t} \cdot \pi_{\text{ct}};\quad \pi_{\text{key}}) \\
\mathcal{D}_{20} &\leftarrow (C,\quad u_2 \cdot \alpha_{\text{key}},\ u_2 \cdot \vec{\gamma}_{\text{key}}) \\
\mathcal{D}_{21} &\leftarrow (C,\quad u_2 \cdot \alpha_{\text{key}} + u_3 \cdot t_0 \cdot \pi_{\text{key}},\ u_2 \cdot \vec{\gamma}_{\text{key}} + u_3 \cdot \vec{t} \cdot \pi_{\text{key}})
\end{aligned}
$$

Then if an algorithm $\mathcal{A}$ distinguishes $\mathcal{D}_{20}$ from $\mathcal{D}_{21}$ with advantage ϵ, then another algorithm $\mathcal{B}$ takes about the same time as $\mathcal{A}$ to break Assumption 2 with advantage ϵ.

The third lemma shows that under Assumption 3, given π_{key} and a randomly tagged version MSK' of the master secret key, it is difficult to distinguish tagged encryption parameters from garbage.

Lemma. *Let $\mathcal{D}_{30}$ and $\mathcal{D}_{31}$ be defined as follows:*

$$\begin{aligned}
(p;\ \mathcal{G}_{\text{key}}, \mathcal{G}_{\text{ct}}, \mathcal{G}_T;\ \otimes;\ \alpha_{\text{ct}}, \alpha_{\text{key}};\ \pi_{\text{ct}}, \pi_{\text{key}}) &\leftarrow \textsc{GroupGenSim}(\lambda) \\
\vec{r} &\xleftarrow{\text{R}} \mathbb{Z}_p^{n+1} \\
(t_0,\ \vec{t}) &\xleftarrow{\text{R}} \mathbb{Z}_p^* \times \mathbb{Z}_p^{n+1} \\
u_1, u_2, r_\delta &\xleftarrow{\text{R}} \mathbb{Z}_p \\
\tau' &\xleftarrow{\text{R}} \mathcal{G}_T \\
\vec{\gamma}_{\text{key}} &\leftarrow \vec{r} \cdot \alpha_{\text{key}} \\
\vec{\gamma}_{\text{ct}} &\leftarrow \vec{r} \cdot \alpha_{\text{ct}} \\
\delta_{\text{key}} &\leftarrow r_\delta \cdot \alpha_{\text{key}}
\end{aligned}$$

$$\begin{aligned}
\text{GP} &\leftarrow (p;\ \mathcal{G}_{\text{key}}, \mathcal{G}_{\text{ct}}, \mathcal{G}_T;\ \otimes) \\
C &\leftarrow (\text{GP};\quad \alpha_{\text{key}}, \vec{\gamma}_{\text{key}};\quad \alpha_{\text{ct}}, \vec{\gamma}_{\text{ct}};\quad \pi_{\text{key}};\ \delta_{\text{key}} + u_2 \cdot \pi_{\text{key}}) \\
\mathcal{D}_{30} &\leftarrow (C,\quad u_1 \cdot \alpha_{\text{ct}} + t_0 \cdot \pi_{\text{ct}},\quad u_1 \cdot \vec{\gamma}_{\text{ct}} + \vec{t} \cdot \pi_{\text{ct}},\quad u_1 \cdot \delta_{\text{key}} \otimes \alpha_{\text{ct}}) \\
\mathcal{D}_{31} &\leftarrow (C,\quad u_1 \cdot \alpha_{\text{ct}} + t_0 \cdot \pi_{\text{ct}},\quad u_1 \cdot \vec{\gamma}_{\text{ct}} + \vec{t} \cdot \pi_{\text{ct}},\quad \tau')
\end{aligned}$$

Then if an algorithm $\mathcal{A}$ distinguishes $\mathcal{D}_{30}$ from $\mathcal{D}_{31}$ with advantage ϵ, then another algorithm $\mathcal{B}$ takes about the same time as $\mathcal{A}$ to break Assumption 3 with advantage ϵ.

Proof. We begin the proof of all 3 assumptions with the same few simplifications.

Rigged group parameters First, we will note that, as above, we can "rig" the group parameters $\alpha_{\text{key}}, \alpha_{\text{ct}}, \beta_{\text{key}}, \beta_{\text{ct}}, \pi_{\text{key}}$ and π_{ct} to whatever we want, so long as they have the correct products. If the lemmas are true with thees rigged group parameters, then by re-randomization we see that they are also true with random group parameters.

Ignoring betas In each assumption we will need only one of β_{key} or β_{ct}. So when rigging the group parameters we will set, eg, $\beta_{\text{key}} = \langle 0, 0, 0, 1 \rangle^\top \cdot \eta$, and confine the other parameters to the first 3 dimensions. The system will then behave as though

β_{key} were completely unused: the 4th coordinate of any key-side element will be uniformly, independently random, and the 4th coordinate of any ct-side element will be 0.

Furthermore, even when we are not ignoring, say, β_{ct}, we can remove β_{ct} terms from any element we wish, because those terms can easily be added back later.

Compression Parameters can be compressed to a few elements. Suppose that the lemmas are true with $n = 0$, so that γ is a scalar rather than a vector. Then choose $f_0, g_0 \xleftarrow{\text{R}} \mathbb{Z}_p$ and $\vec{f}, \vec{g} \xleftarrow{\text{R}} \mathbb{Z}_p^{n+2}$, and set

$$\begin{aligned}
\alpha'_{\text{key}} &\leftarrow f_0 \cdot \alpha_{\text{key}} + g_0 \cdot \gamma_{\text{key}} \\
\vec{\gamma}'_{\text{key}} &\leftarrow \vec{f} \cdot \alpha_{\text{key}} + \vec{g} \cdot \gamma_{\text{key}} \\
u \cdot \alpha'_{\text{ct}} &\leftarrow f_0 \cdot u \cdot \alpha_{\text{ct}} + g_0 \cdot u \cdot \gamma_{\text{ct}} \\
u \cdot \alpha'_{\text{ct}} + t'_0 \cdot \pi'_{\text{ct}} &\leftarrow f_0 \cdot (u \cdot \alpha_{\text{ct}} + t_0 \cdot \pi_{\text{ct}}) + g_0 \cdot (u \cdot \gamma_{\text{ct}} + t \cdot \pi_{\text{ct}}) \\
u \cdot \vec{\gamma}'_{\text{ct}} + \vec{t}' \cdot \pi'_{\text{ct}} &\leftarrow \vec{f} \cdot (u \cdot \alpha_{\text{ct}} + t_0 \cdot \pi_{\text{ct}}) + \vec{g} \cdot (u \cdot \gamma_{\text{ct}} + t \cdot \pi_{\text{ct}})
\end{aligned}$$

and so on. It is easily seen that if $(t_0, t) \cdot \alpha_{\text{key}}$ is linearly independent of $(\alpha_{\text{key}}, \gamma_{\text{key}})$, then the resulting distribution of $u, t'_0, \vec{t}', \alpha'_{\text{key}}, \vec{\gamma}'$ will be uniformly random, and so the resulting parameters will be correctly distributed. Since t_0, t need not be random, we can set $t_0 = 0$.

Extending Assumption 1 Ignore β_{key} and remove β_{ct} terms from α_{ct} and γ_{ct}, and from $u \cdot \alpha_{\text{ct}}$. Set $t_0 = 0$ and

$$\begin{aligned}
\alpha_{\text{key}} &= (1,0,0) \cdot \eta & \alpha_{\text{ct}} &= (s,1,0) \cdot \eta \\
\pi_{\text{ct}} &= (0,0,1) \cdot \eta & \beta_{\text{ct}} &= (0,1,t) \cdot \eta
\end{aligned}$$

We are then given

$$\begin{aligned}
\alpha_{\text{key}} &= (1,0,0) \cdot \eta & \gamma_{\text{key}} &= (r,0,0) \cdot \eta \\
\alpha_{\text{ct}} &= (s,1,0) \cdot \eta & \gamma_{\text{ct}} &= (rs,r,0) \cdot \eta \\
\beta_{\text{ct}} &= (0,1,t) \cdot \eta & u \cdot \alpha_{\text{ct}} &= (su,u,0) \cdot \eta
\end{aligned}$$

and either

$$ru \cdot \alpha_{\text{ct}} + v \cdot \beta_{\text{ct}} = (rsu, ru + v, tv) \cdot \eta$$

or

$$ru \cdot \alpha_{\text{ct}} + v \cdot \beta_{\text{ct}} + t_1 \cdot \pi_{\text{ct}} = (rsu, ru + v, tv + t_1) \cdot \eta$$

where $tv + t_1$ is uniformly random. Thus, given

$$(1, r, s, rs, t, u, su, rsu, ru + v) \cdot \eta$$

we must distinguish $tv \cdot \eta$ from random, which is difficult under Assumption 1. Alternatively, set $u = r, s = 1/r$ and adjust η and t by factors of r to clear denominators. This is legitimate because u can be re-randomized additively, and s can be re-randomized multipliciatively. Then it suffices that given

$$(1, r, r^2, r^3 + rv, t) \cdot \eta$$

it should be hard to distinguish $tv \cdot \eta$ from random.

Extending Assumption 2 Ignore β_{ct}, set $t_0 = 0$ and remove β_{key} components of $\alpha_{\text{key}}, \gamma_{\text{key}}$ and $u \cdot \alpha_{\text{key}}$. Set

$$\begin{array}{lcllcl} \alpha_{\text{key}} & = & (s, -1, 0) \cdot \eta & \alpha_{\text{ct}} & = & (1, 0, 0) \cdot \eta \\ \pi_{\text{key}} & = & (0, 1, 0) \cdot \eta & \pi_{\text{ct}} & = & (1, s, v) \cdot \eta \\ \beta_{\text{key}} & = & (0, -vt, st) \cdot \eta & & & \end{array}$$

where $s, v \xleftarrow{R} \mathbb{Z}_p$. Furthermore, fix the coefficient of β_{key} on the $ru \cdot \gamma_{\text{key}}$ or $ru \cdot \gamma_{\text{key}} + t \cdot \pi_{\text{key}}$ to $1/t$, and set $u_1 = u_2 = u$. We are then given

$$\begin{aligned}
\alpha_{\text{key}} &= (s, -1, 0) \cdot \eta & \gamma_{\text{key}} &= (rs, -r, 0) \cdot \eta \\
\alpha_{\text{ct}} &= (1, 0, 0) \cdot \eta & \gamma_{\text{ct}} &= (r, 0, 0) \cdot \eta \\
\beta_{\text{key}} &= (0, -vt, st) \cdot \eta & \pi_{\text{key}} &= (0, 1, 0) \cdot \eta \\
u \cdot \alpha_{\text{ct}} &= (u, 0, 0) \cdot \eta & u \cdot \gamma_{\text{ct}} + t \cdot \pi_{\text{ct}} &= (ru + t, st, vt) \cdot \eta \\
u \cdot \alpha_{\text{key}} &= (us, -u, 0) \cdot \eta
\end{aligned}$$

and either

$$ur \cdot \alpha_{\text{key}} + \beta_{\text{key}}/t = (urs, -ur - v, s)$$

or

$$ur \cdot \alpha_{\text{key}} + \beta_{\text{key}}/t + u_3 \cdot t \cdot \pi_{\text{key}} = (urs, -ur - v + u_3 \cdot t, s)$$

where $-ur - v + u_3 \cdot t$ is uniformly random. Negating elements as needed for simplicity, we are given

$$(1, r, s, rs, u, ru + t, su, rsu, st, tv) \cdot \eta$$

and wish to distinguish $(ru + v) \cdot \eta$ from random. Once again, if we set $u = r$ and $s = 1/r$ and normalize η and v, we will need to distinguish $(r^3 + v) \cdot \eta$ from random given

$$(1, r, r^2, r^3 + rt, t, tv) \cdot \eta$$

Extending Assumption 3 Ignore both β_{key} and β_{ct}, and set $t = 0$ (instead of $t_0 = 0$ as usual), and furthermore set $r_\delta = t_0$. Set

$$\begin{aligned}
\alpha_{\text{key}} &= (1, s) \cdot \eta & \alpha_{\text{ct}} &= (1, 0) \cdot \eta \\
\pi_{\text{key}} &= (0, 1) \cdot \eta & \pi_{\text{ct}} &= (-s, 1) \cdot \eta
\end{aligned}$$

We are then given

$$\begin{aligned} \alpha_{\text{key}} &= (1, s) \cdot \eta & \gamma_{\text{key}} &= (r, rs) \cdot \eta \\ \alpha_{\text{ct}} &= (1, 0) \cdot \eta & \gamma_{\text{ct}} &= (r, 0) \cdot \eta \\ \pi_{\text{key}} &= (0, 1) \cdot \eta & \delta_{\text{key}} + u_2 \cdot \pi_{\text{key}} &= (t_0, st_0 + u_2) \cdot \eta \\ u_1 \cdot \alpha_{\text{ct}} + t_0 \cdot \pi_{\text{ct}} &= (u_1 - st_0, t_0) \cdot \eta & u_1 \cdot \gamma_{\text{ct}} &= (ru_1, 0) \cdot \eta \end{aligned}$$

and are asked to distinguish

$$u_1 \cdot \delta_{\text{key}} \otimes \alpha_{\text{ct}} = u_1 t_0 \cdot \eta \otimes \eta$$

from random. Note that $st_0 + u_2$ is uniformly random and can be ignored. Collecting terms and negating u_1, we wish to distinguish $u_1 t_0 \cdot \eta \otimes \eta$ from random given

$$(1, r, s, t_0, rs, ru_1, u_1 + st_0)$$

which is difficult under Assumption 3.

This completes the proof of the lemmas. □

We are now ready to dive into the security games. Let w_i denote the probability that the adversary wins Game_i, and let q be the number of queries the adversary makes.

$\text{Game}_{\text{real}}$ This is the real security game. By definition,

$$\text{Adv}(\mathcal{A} \leftrightarrow \text{Game}_{\text{real}}) = |2 \cdot w_i - 1|$$

Game_0 For this game, the simulator replaces the challenge ciphertext with a semi-functional one of tag $(t_0, \vec{t})$. To do this, the simulator replaces the ordinary SETUP with one that samples from $\mathcal{D}_{11}$, and uses the resulting GP, KBP and EP (generating δ_{key} and τ itself). To construct the challenge ciphertext, the simulator uses the "tagged encryption parameters"

$$(u \cdot \alpha_{\text{ct}} + t_0 \cdot \pi_{\text{ct}}, \quad u \cdot \vec{\gamma}_{\text{ct}} + \vec{t} \cdot \pi_{\text{ct}}, \quad \delta_{\text{key}} \otimes (u \cdot \alpha_{\text{ct}} + t_0 \cdot \pi_{\text{ct}}))$$

in place of EP. Note that without the t_0 and $\vec{t}$ terms, as would occur if we sampled from $\mathcal{D}_{10}$ instead of $\mathcal{D}_{11}$, the result has the same distribution as in the real game. Thus there is an algorithm $\mathcal{B}_1$, running in about the same time as $\mathcal{A}$, such that

$$\mathrm{Adv}(\mathcal{B}_1 \leftrightarrow \text{Assumption 1}) = |w_0 - w_{\mathrm{real}}|$$

Game$_i$ This game is the same as Game$_0$, except that the first i delegation queries are answered with semi-functional keys, with uniformly independently random tags (thus Game$_0$ is Game$_i$ with $i = 0$).

To move from Game$_i$ to Game$_{i+1}$, we perform a simulation based on Assumption 2. The simulator samples from either $\mathcal{D}_{20}$ or $\mathcal{D}_{21}$. Once again, it simulates SETUP using the given $\alpha_{\mathrm{key}}, \vec{\gamma}_{\mathrm{key}}, \alpha_{\mathrm{ct}}$ and $\vec{\gamma}_{\mathrm{ct}}$, generating its own δ_{key} and generates the challenge ciphertext from $u \cdot \alpha_{\mathrm{ct}} + t_0 \cdot \pi_{\mathrm{ct}}$, $u \cdot \vec{\gamma}_{\mathrm{ct}} + \vec{t} \cdot \pi_{\mathrm{ct}}$, $\delta_{\mathrm{key}} \otimes (u \cdot \alpha_{\mathrm{ct}} + t_0 \cdot \pi_{\mathrm{ct}})$. It generates the first i secret keys using the real KBP $= (\alpha_{\mathrm{key}}, \vec{\gamma}_{\mathrm{key}})$, but then adds random multiples of π_{key} to each element, thereby producing a random tag. It generates the last $q - i - 1$ keys normally.

The simulator generates the $i + 1$st key using either

$$(u_2 \cdot \alpha_{\mathrm{key}},\ u_2 \cdot \vec{\gamma}_{\mathrm{key}})$$

in place of KBP if it sampled from $\mathcal{D}_{20}$, or

$$(u_2 \cdot \alpha_{\mathrm{key}} + t_0 \cdot \pi_{\mathrm{key}},\ u_2 \cdot \vec{\gamma}_{\mathrm{key}} + \vec{t} \cdot \pi_{\mathrm{key}})$$

in place of KBP if it sampled from $\mathcal{D}_{21}$. Thus in the former case, the $i + 1$st key is fully-functional, whereas in the latter case it is semi-functional with the same tag $(t_0, \vec{t})$ as the challenge ciphertext. However, note that the entire tag is not used for the challenge ciphertext. Rather, the challenge ciphertext uses

$$u_1 \cdot t_0 \quad \text{and} \quad u_1 \cdot \vec{v}^{\top} \cdot \vec{t}$$

and the $i+1$st secret key uses

$$u_2 \cdot t_0 \quad \text{and} \quad u_2 \cdot M^\top \cdot \vec{t}$$

We claim that these are distributed the same as

$$u_1 \cdot t_0,\ u_1 \cdot \vec{v}^\top \cdot \vec{t};\quad u_3 \cdot t_0',\ u_3 \cdot M^\top \cdot \vec{t'}$$

where $(t_0', \vec{t'}) \xleftarrow{\mathrm{R}} \mathbb{Z}_p^* \times \mathrm{Aff}\left(\mathbb{Z}_p^n\right)$ is another uniformly random tag. Thus if the simulator sampled from $\mathcal{D}_{21}$, the adversary will be playing Game_{i+1}. To see this, note that the $i+1$st secret key does not decrypt the challenge ciphertext (or else it would not be a legal query), so $\vec{v}$ is linearly independent from the columns of M. Thus the projection of $(\vec{v} \mid M)^\top \cdot (\vec{t}/t_0)$ is uniformly random in $\mathbb{Z}_p^{d+2}$, where d is the dimension of the subspace $\mathfrak{r}$, and so the two distributions are the same as claimed.

Thus there is an algorithm $\mathcal{B}_2$, running in about the same time as $\mathcal{A}$, such that

$$\mathrm{Adv}(\mathcal{B}_2 \leftrightarrow \text{Assumption 2}) = |w_{i+1} - w_i|$$

Note that in Game_q, the adversary sees only semi-functional secret keys and semi-functional ciphertexts, all with uniformly, independently random tags.

$\mathbf{Game_{final}}$ In the final game, the ciphertext's MKEY is replaced with a random element of $\mathcal{G}_T$. Thus the adversary is carrying out an attack on the underlying symmetric encryption system $\mathbf{E}$, so there is an adversary $\mathcal{B}_{\mathbf{E}}$ such that

$$\mathrm{Adv}(\mathcal{B}_{\mathbf{E}} \leftrightarrow \mathbf{E}) = |2 \cdot w_{\mathrm{final}} - 1|$$

To transition between Game_q and $\mathrm{Game}_{\mathrm{final}}$, we use Assumption 3. The simulator samples from $\mathcal{D}_{30}$ or $\mathcal{D}_{31}$, and uses its EP and KBP as the public parameters, computing $\delta_{\mathrm{key}} \otimes \alpha_{\mathrm{ct}} = (\delta_{\mathrm{key}} + u_3 \cdot \pi_{\mathrm{key}}) \otimes \alpha_{\mathrm{ct}}$, and creating a master key MSK' tagged with $(0, \langle u_3, 0, \ldots, 0\rangle^\top)$ from $\alpha_{\mathrm{key}}, \vec{\gamma}_{\mathrm{key}}$ and $\delta_{\mathrm{key}} + u_3 \cdot \pi_{\mathrm{key}}$. To respond to a delegation query, it passes the tagged MSK' to DELEGATE, which produces a tagged secret key,

then re-randomizes the tag using π_{key}. To produce the challenge ciphertext, the simulator uses the last part of the sample, which is either a tagged copy of the encryption parameters or complete garbage. In the former case, this is identical to Game_q; in the latter case, it is $\text{Game}_{\text{final}}$. Thus there is an algorithm $\mathcal{B}_3$, running in about the same time as $\mathcal{A}$, such that

$$\text{Adv}(\mathcal{B}_3 \leftrightarrow \text{Assumption 3}) = |w_{\text{final}} - w_q|$$

Summing these equations, we find that

$$\begin{aligned}\text{Adv}(\mathcal{A} \leftrightarrow \mathbf{Sp}) \quad \leq \quad & 2 \cdot \text{Adv}(\mathcal{B}_1 \leftrightarrow \text{Assumption 1}) + 2q \cdot \text{Adv}(\mathcal{B}_2 \leftrightarrow \text{Assumption 2}) \\ & +2 \cdot \text{Adv}(\mathcal{B}_3 \leftrightarrow \text{Assumption 3}) + \text{Adv}(\mathcal{B}_{\mathbf{E}} \leftrightarrow \mathbf{E})\end{aligned}$$

as claimed. This completes the proof of Theorem 5.

Composite-order groups We can perform this same proof using composite-order groups instead of prime-order ones, as in [26] and [3]. The three assumptions are the same as in [26] and the lemmas are the same as the ones we proved.[2]

Composite-order groups make things more complicated because spatial encryption requires p to be prime. However, the applications listed in Chapter 4 do not require precise knowledge of p. In particular, consider instantiating each application mod several primes $\{p_i\}$. Then doing the same computations mod $N = \prod_i p_i$ will enforce the delegation and decryption relations mod each prime p_i, even though the primes p_i are unknown.

Multiple masters As before, spatial encryption with the groups used in this proof can be instantiated with multiple masters. The resulting system has the same threshold security properties as before, and it is still difficult to tell which master is being used. However, the more complex groups prevent generation of the public parameters using a random-oracle hash, so a trusted third party is required.

[2] Actually, the third assumption in [26] is slightly stronger than necessary, both for that work and for this one.

Chapter 7

Conclusions and Future Work

We have presented a simple model, GIBE, which describes most variants of identity-based encryption. We have formalized the process by which some GIBEs can be implemented using other GIBEs, and by which signatures can be built from encryption schemes.

Adding to the toolbox of flexible IBE variants, we have introduced two new GIBE systems: spatial encryption and doubly-spatial encryption. The former is more efficient and supports a stronger proof of security, but the latter is more flexible. We have studied how to use these systems to build new encryption schemes, and how to emulate already-known ones with efficiency comparable to single-purpose constructions. By studying sparse products of such systems, we have laid the foundation for an expressive language to describe their roles and policies.

Still, we have only scratched the surface in this paper, leaving plenty of room for future work.

Incremental improvements We would like a proof of adaptive security for doubly-spatial encryption, ideally one which uses more a more natural assumption such as the Decision Linear or Decision Subgroup assumption. We would also like to see variants of spatial encryption which are more than weakly anonymous.

Tensors Doubly-spatial encryption hints at a more general form of spatial encryption in which roles and policies are tensor spaces. We are interested both in whether such a system can be built securely and efficiently, and whether it will enable us to implement encryption systems for new and important hierarchies. We are especially hopeful given the similarities between spatial encryption and other identity-based encryption technologies.

Similarly, we wonder what other GIBEs other than (doubly) spatial encryption and attribute-based encryption could be used to form new and powerful hierarchies.

Implementation We have begun implementing spatial encryption, but did not finish it for this paper due to time pressure. Perhaps the most interesting aspect of such an implementation would be its policy language; this we have also laid the foundation for, but not completed.

We hope that this work, and future work in the same direction, will make it easier to design, build and deploy modern cryptographic systems.

Bibliography

[1] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[2] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In *CRYPTO*, pages 98–115, 2010.

[3] Nuttapong Attrapadung and Benoît Libert. Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. *Public Key Cryptography–PKC 2010*, pages 384–402, 2010.

[4] Dan Bernstein. Breaking DNSSEC, 2009 . http://cr.yp.to/talks/2009.10.30/slides.pdf.

[5] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2007.

[6] D. Boneh and X. Boyen. Short signatures without random oracles. In *Advances in Cryptology–Eurocrypt 2004*, pages 56–73. Springer, 2004.

[7] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology–Eurocrypt 2005*. Springer-Verlag.

[8] Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology–Crypto 2001*, pages 213–229. Springer, 2001.

[9] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *Foundations of Computer Science–FOCS 2007*, pages 647–657. IEEE, 2007.

[10] Dan Boneh and Michael Hamburg. Generalized identity based and broadcast encryption schemes. In *Advances in Cryptology–Asiacrypt 2008*, Asiacrypt '08, pages 455–470. Springer-Verlag, 2008.

[11] Xavier Boyen. Multipurpose identity-based signcryption. *Advances in Cryptology–Crypto 2003*, pages 383–399, 2003.

[12] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). *Advances in Cryptology–Crypto 2006*, pages 290–307, 2006.

[13] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *Advances in Cryptology-Eurocrypt 2003*, pages 646–646, 2003.

[14] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology–Eurocrypt 2004*, pages 207–222. Springer, 2004.

[15] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.

[16] Clifford Cocks. An identity based encryption scheme based on quadratic residues. *Cryptography and Coding*, pages 360–363, 2001.

[17] Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano. Hidden vector encryption fully secure against unrestricted queries: No question left unanswered, 2011.

[18] Leo Ducas. Anonymity from asymmetry: New constructions for anonymous hibe. *Topics in Cryptology-CT-RSA 2010*, pages 148–164, 2010.

[19] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[20] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. *Advances in Cryptology-Asiacrypt 2002*, pages 149–155, 2002.

[21] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. of ACM-CCS 2006*, CCS '06, pages 89–98. ACM.

[22] Jeremy Horwitz and Benjamin Lynn. Toward hierarchical identity-based encryption. In *Advances in Cryptology-Eurocrypt 2002*, pages 466–481. Springer, 2002.

[23] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology–Eurocrypt 2008*, pages 146–162. Springer-Verlag, 2008.

[24] Ben Laurie, Geoffrey Sisson, Roy Arends, and David Blacka. DNS security (DNSSEC) hashed authenticated denial of existence, 2008. RFC 5155.

[25] Allison Lewko, Amit Sahai, Tatsuaki Okamoto, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption.

[26] Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. Cryptology ePrint Archive, Report 2009/482, 2009.

[27] Silvio Micali, Michael Rabin, and Joe Kilian. Zero-knowledge sets. In *Foundations of Computer Science–FOCS 2003*, pages 80–91. IEEE, 2003.

[28] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. *Advances in Cryptology–Asiacrypt 2009*, pages 214–231, 2009.

[29] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *Proc. of ACM-CCS 2007*, pages 195–203. ACM, 2007.

[30] Ronald Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology–Asiacrypt 2001*, pages 554–567. Springer-Verlag, 2001.

[31] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. *Advances in Cryptology–Eurocrypt 2005*, pages 457–473, 2005.

[32] Ryuichi Sakai and Jun Furukawa. Identity-based broadcast encryption, 2007.

[33] Jakob Schlyter. DNS security (DNSSEC) NextSECure (NSEC) RDATA format, 2004. RFC 3845.

[34] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[35] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology–Crypto'84*, 1984.

[36] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. *Theory of Cryptography*, pages 457–473, 2009.

[37] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *Advances in Cryptology–Crypto 2009*, Lecture Notes in Computer Science. Springer Berlin / Heidelberg.

[38] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *Proc. of ACM-CCS 2004*, pages 354–363. ACM, 2004.