A Note on Inner-Product Secret-Key ABE

Maya Kalai and Ella Kim

MIT PRIMES

Abstract. Consider a scenario where Alice would like to encrypt a message such that Bob and Charlie can decrypt it if and only if they have an attribute satisfying a given predicate. This problem is solved by attribute-based encryption and has applications in many areas, including electronic record keeping and the Internet of Things.

In this note, we construct and implement a secret-key attribute-based encryption (SK-ABE) scheme supporting inner-product predicates, purely in the random oracle model. Our construction works by adapting the work of Attrapadung et al. (Crypto '18) building secret-key ABE from constrained PRFs and the recent work of Servan-Schreiber (Asiacrypt '24) building constrained pseudorandom functions from random oracles.

Our construction is practical, which we demonstrate with a prototype implementation. With attribute vectors of length 1,000, each encryption and decryption operation takes approximately 250 microseconds, and scales linearly with the length of the attribute vectors.

1 Introduction

Attribute-Based Encryption (ABE) [SW05] is a type of encryption in which the ability to decrypt a ciphertext depends on whether the user's attributes satisfy a given criteria (or "predicate"). Instead of basing access solely on a user's identity [Sha84], ABE allows decryption if and only if the required attributes—such as having a certain role, holding a specific rank, or being above a certain age—are present in the ciphertext. More specifically, each user is associated with a set of attributes and is given a secret key associated with these attributes which is computed from a master secret key. Each ciphertext is a function of the message to be encrypted, the master secret key, and the attributes required to decrypt.

Inner-Product Attribute-Based Encryption (IP-ABE) is a specialized form of ABE where the decryption condition is expressed as an inner-product predicate. A user can decrypt a given ciphertext if and only if the inner-product of the user's attribute vector and the ciphertext's attribute vector meets the required criterion (for example, if the inner-product equals zero).

To understand the practical implications, consider a hospital that encrypts sensitive patient records. Each encrypted file can be associated with attributes such as "cardiology," "2024," and "surgery." The user's decryption key might correspond to a vector of attributes representing their department, clearance level, or assigned roles. Under an IP-ABE scheme, only those with the correct combination of attributes—captured mathematically by the inner product test—would be able to decrypt and access the protected information.

More specifically, suppose each hospital employee is associated with a vector of attributes, $\mathbf{a} \in \{0,1\}^k$ such that $a_i = 0$ if and only if the employee has attribute i. A hospital employee wants to encrypt a message that can only be decrypted by cardiology surgeons. To address this task, we can create an IP-ABE scheme where we assign vectors to the message and hospital employees. The message vector would have 1's in the coordinates corresponding to "surgeon" and "cardiology," and all other coordinates equal to 0. A surgeon in the cardiology department would have a vector where the 0's are in the coordinates that correspond to "surgeon" and "cardiology," and all other coordinates are equal to 1. This way, the inner-product will be zero (i.e., the predicate will be satisfied) if and only if the necessary attributes for access are satisfied.

We could further adapt this vector to allow for additional attributes, like a surgeon in both the cardiology and pediatrics departments. This surgeon would have a vector where the 0's are in the coordinates that correspond to "surgeon," "cardiology," and "pediatrics," and all other coordinates are equal to 1. The scheme would work because the inner-product of the cardiology/pediatrics surgeon's vector and the message vector still equals zero.

1.1 Technical Overview

In this section, we briefly describe the results in this note. At a high level, we instantiate the framework of Attrapadung et al. [AMN+18] using the constrained pseudorandom function (CPRF) of Servan-Schreiber [SS24] built from random oracles.

The SK-ABE framework of Attrapadung et al. [AMN+18]. This framework uses two ingredients: a CPRF scheme and a symmetric-key encryption scheme.

Tool: Constrained PRFs. Constrained pseudorandom functions (CPRFs) are PRFs that support the ability to create constrained keys, which are keys that allow the holder to evaluate the PRF on a subset of inputs in the PRF domain while being unable to evaluate the PRF on other inputs. Which inputs are "constrained" is determined by a predicate, and in this note we will explicitly focus on inner-product predicates. We will use the recent CPRF construction of Servan-Schreiber [SS24]. A symmetric-key encryption scheme uses the same secret key for both encrypting and decrypting data. It ensures confidentiality by transforming plaintext into ciphertext that can only be reversed with the correct key. Known for its speed and efficiency, symmetric-key encryption is widely used in applications requiring high performance, such as securing communication channels or storing sensitive data. Popular algorithms include AES (Advanced Encryption Standard).

Tool: Symmetric-key Encryption. A symmetric-key encryption scheme uses the same secret key for both encrypting and decrypting messages. Symmetric-key encryption can be built from the minimal assumption that one-way functions exist, which is trivially implied by a random oracle.

Building SK-ABE from CPRFs. The idea is to realize SK-ABE from a CPRF stems from the observation that a constrained key can be used to derive a symmetric-key encryption key.

Using the master secret key (which consists of the master CPRF key) the encryptor can generate a constrained secret key for the CPRF, for each user that matches up with their attribute. In this way, they can evaluate the CPRF if and only if their attribute is satisfied and derive a decryption key if and only if the attribute satisfies the predicate. In more detail:

- The master key of the SK-ABE scheme corresponds to the CPRF master key msk.
- To encrypt a message with attribute $\mathbf{x}$, the encryptor first derives a secret key k by evaluating the CPRF on input $\mathbf{x}$ using the CPRF master key msk.
- To generate a key for a particular predicate $\mathbf{y}$, the encryptor generates a CPRF constraint key csk with a predicate vector $\mathbf{y}$.
- To decrypt a ciphertext $\mathsf{ct}_{\mathbf{x}}$ encrypted with attribute $\mathbf{x}$, the decryptor evaluates the CPRF using the constrained key csk to derive a key k' and uses the symmetric-key encryption algorithm to decrypt $\mathsf{ct}_{\mathbf{x}}$.

The idea is that if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ (predicate satisfied), then $k' = k$ which allows the decryptor to recover the message.

Remark 1 (Arbitrary CPRFs). While we focus on the random oracle based CPRF construction, this template works for arbitrary CPRFs including the DDH-based construction for inner-products constructed by Servan-Schreiber [SS24].

Remark 2 (Predicate Privacy). If the CPRF satisfies constraint privacy, then the SK-ABE sketched above also satisfies predicate privacy. In words, predicate privacy states that the SK-ABE key does not reveal the predicate $\mathbf{y}$, which follows straightforwardly from the CPRF constraint-hiding property.

Concretely-efficient SK-ABE? Because the CPRF construction of Servan-Schreiber is concretely efficient, in this note we show that this efficiency translates to the SK-ABE scheme. In Section 3, we show that our implementation achieves concretely efficient performance even with long attribute vectors.

2 Construction

We first describe the necessary preliminaries for the construction in Section 2.1. Our construction is presented in Section 2.2

2.1 Preliminaries

2.1.1 Notation We let $\mathbb{N}$ denote the set of natural numbers and $\mathbb{F}$ denote a finite field (e.g., integers mod p). We denote by $\mathsf{poly}(\cdot)$ the set of all polynomials and by $\mathsf{negl}(\cdot)$ any negligible function.

Vectors and matrices. A vector $\mathbf{v} = (v_1, \ldots, v_n)$ is denoted using bold lowercase letters. The inner product between two vectors $\mathbf{a}$ and $\mathbf{b}$ is denoted $\langle \mathbf{a}, \mathbf{b} \rangle$.

Sampling and assignment. We let $x \xleftarrow{R} S$ denote a uniformly random sample drawn from a set S. We let $x \leftarrow \mathcal{A}$ denote assignment from a randomized algorithm $\mathcal{A}$ and $x := y$ denote initialization of x to the value of y (which may be the output of a deterministic algorithm).

Efficiency and indistinguishability. By an *efficient* algorithm $\mathcal{A}$ we mean that $\mathcal{A}$ is modeled by a (possibly non-uniform) Turing Machine that runs in probabilistic polynomial time. We write $D_0 \approx_c D_1$ to mean that two distributions D_0 and D_1 are *computationally* indistinguishable to all efficient distinguishers $\mathcal{D}$ and $D_0 \approx_s D_1$ to mean that D_0 and D_1 are *statistically* indistinguishable.

2.1.2 Attribute-based encryption We recall the definition of attribute-based encryption. Our definition is tailored to the case of inner-product predicates and 1-key, selective security. This corresponds to the primitive we construct in Section 2.2.

Definition 3 (Inner-Product Attribute Based Encryption). *Let $\lambda \in \mathbb{N}$ be a security parameter and let $\mathbb{F}$ be a field. An inner-product secret-key attribute-based encryption scheme (IP-SK-ABE) scheme over $\mathbb{F}$ is associated with the following four efficient algorithms:*

- $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{msk}$. *The randomized setup algorithm takes as input the security parameter λ and a vector length n. It outputs a master secret key* msk.
- $\mathsf{Enc}(\mathsf{msk}, \mathbf{x}, m) \to \mathsf{ct}_\mathbf{x}$. *The randomized encryption algorithm takes as input a master secret key* msk*, an attribute $\mathbf{x} \in \mathbb{F}^n$, and a message $m \in \mathbb{F}$. It outputs a ciphertext* $\mathsf{ct}_\mathbf{x}$.
- $\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y}) \to \mathsf{sk}_\mathbf{y}$. *The randomized key generation algorithm takes as input a master secret key* msk *and the predicate vector $\mathbf{y} \in \mathbb{F}^n$. It outputs a secret key* $\mathsf{sk}_\mathbf{y}$.
- $\mathsf{Dec}(\mathsf{sk}_\mathbf{y}, \mathsf{ct}_\mathbf{x}) \to m$ *or* $\perp$. *The deterministic decryption algorithm takes as input a secret key* $\mathsf{sk}_\mathbf{y}$ *and ciphertext* $\mathsf{ct}_\mathbf{x}$. *It outputs a message m if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ and $\perp$ otherwise.*

We let any auxiliary public parameters PP *be an implicit input to all algorithms. The IP-SK-ABE functionality must satisfy the following properties:*

Correctness. *For all security parameters $\lambda \in \mathbb{N}$, for all vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}^n$, and for all messages $m \in \mathbb{F}$ it holds that:*

- *If $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ (predicate is satisfied), then*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_\mathbf{y}, \mathsf{ct}_\mathbf{x}) = m : \begin{array}{r} \mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \mathsf{ct}_\mathbf{x} \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathbf{x}, m) \\ \mathsf{sk}_\mathbf{y} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

- *If $\langle \mathbf{x}, \mathbf{y} \rangle \neq 0$ (predicate is not satisfied), then*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_\mathbf{y}, \mathsf{ct}_\mathbf{x}) = \perp : \begin{array}{r} \mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \mathsf{ct}_\mathbf{x} \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathbf{x}, m) \\ \mathsf{sk}_\mathbf{y} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

(1-key, selective) Security. *A SK-ABE scheme is (1-key, selectively)-secure if for all efficient adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following security experiment $\mathsf{Exp}^{\mathsf{abe}}_{\mathcal{A},b}(\lambda)$ is negligible in λ. Here, b denotes the challenge bit.*

1. ***Setup:*** *On input* 1^λ*, the challenger runs* $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ *and then runs* $\mathcal{A}(\mathsf{msk})$.
2. ***Key generation query:*** $\mathcal{A}$ *sends the vector* $\mathbf{y} \in \mathbb{F}^n$ *to the challenger before issuing any other queries. The challenger computes* $\mathsf{sk}_\mathbf{y} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y})$*, and sends* $\mathsf{sk}_\mathbf{y}$ *to* $\mathcal{A}$.
3. ***Encryption queries:*** $\mathcal{A}$ *adaptively sends arbitrary input tuples of the form* $(\mathbf{x}, m_0, m_1)$ *to the challenger, where* $\mathbf{x} \in \mathbb{F}^n$ *subject to* $\langle \mathbf{x}, \mathbf{y} \rangle \neq 0$*. For each such tuple, the challenger computes* $\mathsf{ct}_\mathbf{x} \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathbf{x}, m_b)$ *and sends* $\mathsf{ct}_\mathbf{x}$ *to* $\mathcal{A}$.
4. ***Guess:*** $\mathcal{A}$ *outputs its guess* b'*, which is the output of the experiment.*

$\mathcal{A}$ *wins if* $b' = b$*, and its advantage* $\mathsf{Adv}^{\mathsf{abe}}_{\mathcal{A}}(\lambda)$ *is defined as*

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

where the probability is over the randomness of $\mathcal{A}$*,* Setup*,* KeyGen*, and* Enc.

Definition 4 (Predicate Privacy). *An SK-ABE scheme is (1-key, selectively)-predicate-hiding if for all efficient adversaries* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in the following security experiment* $\mathsf{Exp}^{\mathsf{priv}}_{\mathcal{A},b}(\lambda)$ *is negligible in* λ*. Here,* b *denotes the challenge bit.*

1. ***Setup:*** *On input* 1^λ*, the challenger runs* $\mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda, 1^n)$ *and runs* $\mathcal{A}(\mathsf{msk})$.
2. ***Key generation query:*** $\mathcal{A}$ *sends two vectors* $\mathbf{y}_0, \mathbf{y}_1 \in \mathbb{F}^n$ *to the challenger before issuing any other queries. The challenger computes* $\mathsf{sk}_{\mathbf{y}_b} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathbf{y}_b)$*, and sends* $\mathsf{sk}_{\mathbf{y}_b}$ *to* $\mathcal{A}$.
3. ***Encryption queries:*** $\mathcal{A}$ *adaptively sends arbitrary input tuples of the form* $(\mathbf{x}, m)$ *to the challenger, where* $\mathbf{x} \in \mathbb{F}^n$ *subject to* $\langle \mathbf{x}, \mathbf{y}_0 \rangle = \langle \mathbf{x}, \mathbf{y}_1 \rangle$*. For each such tuple, the challenger computes* $\mathsf{ct}_\mathbf{x} \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathbf{x}, m)$ *and sends* $\mathsf{ct}_\mathbf{x}$ *to* $\mathcal{A}$.
4. ***Guess:*** $\mathcal{A}$ *outputs its guess* b'*, which is the output of the experiment.*

$\mathcal{A}$ *wins if* $b' = b$ *and its advantage* $\mathsf{Adv}^{\mathsf{priv}}_{\mathcal{A}}(\lambda)$ *is defined as*

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

where the probability is over the randomness of $\mathcal{A}$*,* Setup*,* KeyGen*, and* Enc.

2.1.3 Constrained pseudorandom functions We recall the syntax and properties of CPRFs. We restrict the definition to 1-key, constraint-hiding CPRFs for inner-product predicates, which is what we will need to instantiate the construction in Section 2.2.

Definition 5 (Constrained Pseudorandom Functions). *Let* $\lambda \in \mathbb{N}$ *be a security parameter. A Constrained Pseudorandom Function (CPRF) supporting inner-product predicates over a field* $\mathbb{F}$ *and output range* $\mathcal{Y}$ *consists of four efficient algorithm:*

- $\mathsf{KeyGen}(1^\lambda, 1^n) \to \mathsf{msk}$*. The randomized key generation algorithm takes as input a security parameter* λ *and vector length* n*. It outputs a master secret key* msk*.*
- $\mathsf{Eval}(\mathsf{msk}, \mathbf{x}) \to y$*. The deterministic evaluation algorithm takes as input the master secret key* msk *and input* $\mathbf{x} \in \mathbb{F}^n$*. It outputs* $y \in \mathcal{Y}$*.*
- $\mathsf{Constrain}(\mathsf{msk}, \mathbf{z}) \to \mathsf{csk}$*. The randomized constrain algorithm takes as input the master secret key* msk *and a constraint vector* $\mathbf{z} \in \mathbb{F}^n$*. It outputs a constrained key* csk*.*
- $\mathsf{CEval}(\mathsf{csk}, \mathbf{x}) \to y$*. The deterministic constrained evaluation algorithm takes as input the constrained key* csk *and an input* $\mathbf{x} \in \mathbb{F}^n$*. It outputs* $y \in \mathcal{Y}$*.*

We let any auxiliary public parameters PP *be an implicit input to all algorithms. A CPRF must satisfy the following correctness and security properties.*

Correctness. *For all security parameters* λ*, all constraint vectors* $\mathbf{z} \in \mathbb{F}^n$*, and all inputs* $x \in \mathbb{F}^n$ *such that* $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ *(i.e., authorized), it holds that:*

$$\Pr\left[\mathsf{Eval}(\mathsf{msk}, x) = \mathsf{CEval}(\mathsf{csk}, \mathbf{x}) : \begin{array}{c} \mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \mathsf{csk} \leftarrow \mathsf{Constrain}(\mathsf{msk}, \mathbf{z}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

(1-key, selective) Security. *A CPRF is (1-key, selectively)-secure if for all efficient adversaries* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in the following security experiment* $\mathsf{Exp}^{\mathsf{cprf}}_{\mathcal{A},b}(\lambda)$ *is negligible in* λ*. Here,* b *denotes the challenge bit.*

1. ***Setup:*** *On input* 1^λ*, the challenger runs* $\mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda)$*, initializes* $Q := \emptyset$*, and runs* $\mathcal{A}(1^\lambda)$*.*
2. ***Constrain query:*** $\mathcal{A}$ *sends the constraint* $\mathbf{z} \in \mathbb{F}^n$ *to the challenger before issuing any other queries. The challenger computes* $\mathsf{csk} \leftarrow \mathsf{Constrain}(\mathsf{msk}, \mathbf{z})$*, and sends* csk *to* $\mathcal{A}$*.*
3. ***Pre-challenge queries:*** $\mathcal{A}$ *adaptively sends arbitrary inputs* $\mathbf{x} \in \mathbb{F}^n$ *to the challenger. For each* $\mathbf{x}$*, the challenger computes* $y := \mathsf{Eval}(\mathsf{msk}, \mathbf{x})$*, sends* y *to* $\mathcal{A}$*, and proceeds to update* $Q := Q \cup \{\mathbf{x}\}$*.*
4. ***Challenge query:*** *For the single challenge query,* $\mathcal{A}$ *sends input* $\mathbf{x}^* \in \mathbb{F}^n$ *as its challenge query, subject to the restriction that* $\mathbf{x}^* \notin Q$ *and* $C(\mathbf{x}^*) \neq 0$*. If* $b = 0$*, the challenger computes* $y^* := \mathsf{Eval}(\mathsf{msk}, \mathbf{x}^*)$*. Else, if* $b = 1$*, the challenger samples* $y^* \xleftarrow{R} \mathcal{Y}$*. The challenger sends* y^* *to* $\mathcal{A}$*.*
5. ***Post-challenge queries:*** $\mathcal{A}$ *continues to adaptively query the challenger on inputs* $\mathbf{x} \in \mathbb{F}^n$*, subject to the restriction that* $\mathbf{x} \neq \mathbf{x}^*$*. For each* $\mathbf{x}$*, the challenger computes* $y := \mathsf{Eval}(\mathsf{msk}, \mathbf{x})$ *and sends* y *to* $\mathcal{A}$*.*
6. ***Guess:*** $\mathcal{A}$ *outputs its guess* b'*, which is the output of the experiment.*

$\mathcal{A}$ *wins if* $b' = b$*, and its advantage* $\mathsf{Adv}^{\mathsf{cprf}}_{\mathcal{A}}(\lambda)$ *is defined as*

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

where the probability is over the randomness of $\mathcal{A}$ *and* KeyGen *and* Constrain*.*

Definition 6 (Constraint Privacy). *A CPRF is (1-key, selectively)-constraint-hiding if for all efficient adversaries* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in the following security experiment* $\mathsf{Exp}^{\mathsf{priv}}_{\mathcal{A},b}(\lambda)$ *is negligible in* λ*. Here,* b *denotes the challenge bit.*

1. ***Setup:*** *On input* 1^λ*, the challenger runs* $\mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda)$*, initializes* $Q := \emptyset$*, and runs* $\mathcal{A}(1^\lambda)$*.*
2. ***Constrain query:*** *Before issuing any queries,* $\mathcal{A}$ *sends a* pair *of constraint vectors* $(\mathbf{z}_0, \mathbf{z}_1) \in \mathbb{F}^n \times \mathbb{F}^n$ *to the challenger. The challenger responds with* $\mathsf{csk} \leftarrow \mathsf{Constrain}(\mathsf{msk}, \mathbf{z}_b)$*.*
3. ***Evaluation queries:*** $\mathcal{A}$ *adaptively sends arbitrary input values* $\mathbf{x} \in \mathbb{F}^n$ *to the challenger subject to the restriction that* $\langle \mathbf{z}_0, \mathbf{x} \rangle = \langle \mathbf{z}_1, \mathbf{x} \rangle$*. For each such* $\mathbf{x}$*, the challenger computes* $y := \mathsf{Eval}(\mathsf{msk}, \mathbf{x})$ *and sends* y *to* $\mathcal{A}$*.*
4. ***Guess:*** $\mathcal{A}$ *outputs its guess* b'*, which is the output of the experiment.*

$\mathcal{A}$ *wins if* $b' = b$ *and its advantage* $\mathsf{Adv}^{\mathsf{priv}}_{\mathcal{A}}(\lambda)$ *is defined as*

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

where the probability is over the randomness of $\mathcal{A}$ *and* KeyGen *and* Constrain*.*

Definition 7 (Symmetric-Key Encryption). *A symmetric-key encryption scheme* SKE *consists of three efficient algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax:*

- $\mathsf{KeyGen}(1^\lambda) \to k$*. The randomized key generation algorithm takes as input the security parameter* λ *and outputs a key* $k \in \{0,1\}^\lambda$
- $\mathsf{Enc}(k, m) \to c$*. The randomized encryption algorithm takes a key* k *and message* $m \in \{0,1\}^*$*. It outputs a ciphertext* c
- $\mathsf{Dec}(k, c) \to m$ *or* $\perp$*. The deterministic decryption algorithm takes a key* k *and ciphertext* c*. It either outputs a message* m *or a special symbol* $\perp$

The scheme must satisfy:

Correctness. *For all* $\lambda \in \mathbb{N}$ *and all messages* m*:*

$$\Pr\Big[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m : \ k \leftarrow \mathsf{KeyGen}(1^\lambda)\Big] \geq 1 - \mathsf{negl}(\lambda)$$

Semantic Security. *A symmetric-key encryption scheme* SKE *is semantically secure if for all efficient adversaries* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in the following security experiment* $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},b}(\lambda)$ *is negligible in* λ*. Here,* b *denotes the challenge bit.*

1. ***Setup:*** *On input* 1^λ*, the challenger runs* $k \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and runs* $\mathcal{A}(1^\lambda)$*.*
2. ***Challenge:*** $\mathcal{A}$ *sends a pair of messages* $(m_0, m_1) \in \{0,1\}^* \times \{0,1\}^*$ *of equal length to the challenger. The challenger responds with* $c \leftarrow \mathsf{Enc}(k, m_b)$*.*

3. ***Guess:*** *$\mathcal{A}$ outputs its guess b', which is the output of the experiment.*

$\mathcal{A}$ wins if $b' = b$ and its advantage $\mathsf{Adv}^{\mathsf{sec}}_{\mathcal{A}}(\lambda)$ is defined as

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

where the probability is over the randomness of $\mathcal{A}$, KeyGen, and Enc.

Wrong-Key Rejection. *For any message m and any key k':*

$$\Pr\left[\mathsf{Dec}(k', \mathsf{Enc}(k, m)) \neq \bot : \ k \leftarrow \mathsf{KeyGen}(1^\lambda)\right] \leq \mathsf{negl}(\lambda).$$

In other words, providing the wrong key outputs $\bot$.

Generic upgrades for supporting wrong-key rejection. For any symmetric-key encryption scheme $\mathsf{SKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, we can assume that $\mathsf{SKE.Dec}$ outputs $\bot$ when given the wrong decryption key for a ciphertext. This is without loss of generality because we can either pick SKE to be an authenticated encryption scheme (e.g., encrypt-then-MAC) to ensure $\mathsf{SKE.Dec}$ outputs $\bot$ if given the wrong key for decryption or, alternatively, encrypt each message prefixed with 0^λ and returning $\bot$ if the prefix is not present in the decryption.

2.2 Predicate-hiding SK-ABE construction

Our construction is presented in Figure 1 and closely follows the overview from Section 1.1.

Remark 8. In our construction, we assume that the range $\mathcal{Y}$ of the CPRF corresponds to $\{0,1\}^\lambda$. This is without loss of generality: any range $\mathcal{Y}$ can be made sufficiently large by using $\Omega(\lambda)$ independent CPRF instances. From this augmented range, λ bits of entropy can be extracted using a suitable randomness extractor [ILL89]. This allows us to use the output of the CPRF as the randomness in $\mathsf{SKE.KeyGen}$ to derive a symmetric key.

Predicate-Hiding Inner-Product Secret-Key ABE from CPRFs

Public Parameters. Let $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ be a constraint-hiding CPRF and let $\mathsf{SKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a secret-key encryption scheme as defined in Definition 7.

$\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{msk}$:

1 : $\mathsf{msk}_{\mathsf{cprf}} \leftarrow \mathsf{CPRF.KeyGen}(1^\lambda, 1^n)$
2 : $\mathsf{msk} := \mathsf{msk}_{\mathsf{cprf}}$
3 : **return** msk

$\mathsf{KeyGen}(\mathsf{msk}, \mathbf{y}) \to \mathsf{sk}_{\mathbf{y}}$:

1 : $\mathsf{csk} \leftarrow \mathsf{CPRF.Constrain}(\mathsf{msk}, \mathbf{y})$
2 : $\mathsf{sk}_{\mathbf{y}} := \mathsf{csk}$
3 : **return** $\mathsf{sk}_{\mathbf{y}}$

$\mathsf{Enc}(\mathsf{msk}, \mathbf{x}, m) \to \mathsf{ct}_{\mathbf{x}}$:

1 : $r \leftarrow \mathsf{CPRF.Eval}(\mathsf{msk}, \mathbf{x})$
2 : $k := \mathsf{SKE.KeyGen}(1^\lambda; r)$
3 : $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(k, m)$
4 : $\mathsf{ct}_{\mathbf{x}} = (\mathsf{ct}, \mathbf{x})$
5 : **return** $\mathsf{ct}_{\mathbf{x}}$

$\mathsf{Dec}(\mathsf{sk}_{\mathbf{y}}, \mathsf{ct}_{\mathbf{x}}) \to m$:

1 : **parse** $\mathsf{ct}_{\mathbf{x}} = (\mathsf{ct}, \mathbf{x})$
2 : $r \leftarrow \mathsf{CPRF.CEval}(\mathsf{sk}_{\mathbf{y}}, \mathbf{x})$
3 : $k := \mathsf{SKE.KeyGen}(1^\lambda; r)$
4 : $m \leftarrow \mathsf{SKE.Dec}(k, \mathsf{ct})$
5 : **return** m

Fig. 1: Predicate-Hiding Inner-Product Secret-Key ABE Construction.

2.2.1 Security analysis We analyze the correctness and security of our construction. We note that the security reduces almost directly to the security of the CPRF and symmetric-key encryption scheme.

Correctness. First, observe that $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(k, m)$, where k is derived using randomness output by the CPRF. We analyze the two cases:

- **Case I: Correct key.** First, by correctness of the CPRF, r is the same in both Dec and Enc because we have that $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. Then, because $\mathsf{SKE.KeyGen}$ is computed deterministically using the same randomness in both cases, by the correctness of SKE we have that,

$$\Pr\Big[\mathsf{SKE.Dec}(k, \mathsf{ct}) = m\Big] \geq 1 - \mathsf{negl}(\lambda).$$

  Then, we have that $\Pr[\mathsf{Dec}(\mathsf{sk}_{\mathbf{y}}, \mathsf{ct}_{\mathbf{x}}) = m] \geq 1 - \mathsf{negl}(\lambda)$, as required.
- **Case II: Wrong key.** First, by the pseudorandomness of the CPRF, with all but negligible probability, r will be computed differently in both Dec and Enc, because we have that $\langle \mathbf{x}, \mathbf{y} \rangle \neq 0$. Then, because $\mathsf{SKE.KeyGen}$ is computed deterministically using different randomness, with all but negligible probability Dec and Enc will produce different keys $k \neq k'$. Finally, by the wrong-key rejection property of the SKE scheme,

$$\Pr\Big[\mathsf{SKE.Dec}(k', \mathsf{Enc}(k, m)) \neq \bot\Big] \leq \mathsf{negl}(\lambda).$$

  This implies that $\Pr[\mathsf{Dec}(\mathsf{sk}_{\mathbf{y}}, \mathsf{ct}_{\mathbf{x}}) \neq \bot] \leq \mathsf{negl}(\lambda)$, which is equivalent to $\Pr[\mathsf{Dec}(\mathsf{sk}_{\mathbf{y}}, \mathsf{ct}_{\mathbf{x}}) = \bot] \geq 1 - \mathsf{negl}(\lambda)$, as required.

Security. We now analyze the security of the scheme.

- *Hybrid* $\mathcal{H}_0$. This hybrid consists of the 1-key, selective security game with $b = 0$ and the IP-SKE-ABE scheme presented in Figure 1.
- *Hybrid* $\mathcal{H}_1$. In this hybrid, the challenger replaces the r (as output by CEval in Dec with a uniformly random output in $\mathcal{Y}$). This hybrid is computationally indistinguishable from the previous one by the security of the CPRF.
- *Hybrid* $\mathcal{H}_2$. In this hybrid, we the challenger responds with encryptions of m_1, that is it responds with $\mathsf{ct}_{\mathbf{x}} = (\mathsf{ct}, \mathbf{x})$ where $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(k, m_1)$. This hybrid is computationally indistinguishable from the previous one by the security of the secret key encryption scheme.
- *Hybrid* $\mathcal{H}_3$. In this hybrid, we reverse the changes made in $\mathcal{H}_1$ and generate the randomness using the CPRF. This hybrid is again indistinguishable from the previous one by the security of the CPRF.

This proves that $\mathcal{A}$ has at most negligible advantage in distinguishing between messages m_0 and m_1, which concludes the proof of security.

3 Evaluation

In this section, we benchmark our IP-SK-ABE construction. We implement the construction in Go v1.20 and evaluate our construction using a 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz CPU. Our implementation uses `SHA256` to heuristically instantiate the random oracle (used to implement the CPRF of Servan-Schreiber [SS24]). We use `AES-GCM` to instantiate the symmetric-key encryption scheme. Our implementation is open source: https://github.com/ellajkim/sk-abe.

We analyze the complexity of the CPRF and SKE components of the scheme and report the performance. The construction for the CPRF requires computing the inner product and calling the random oracle, which is instantiated using the `SHA256` hash function. The inner product computation requires a linear number of field additions and multiplications, where the field size is 2^{λ}.

Performance of our IP-SK-ABE scheme. For each encryption and for each decryption, our implementation requires one call to the CPRF and one call to the SKE. We report the performance for varying vector lengths in Table 1. Our measurements show that our scheme's encryption/decryption is less than 1 millisecond for vectors with length less than a few thousand. Therefore, this scheme is practical for real-world applications with a reasonable number of attributes.

(ℓ)	5	10	20	50	100	500	1000	2500	10000	100000
	2 μs	3 μs	5 μs	12 μs	26 μs	132 μs	254 μs	626 μs	2972 μs	26503 μs

Table 1: Concrete evaluation time for vectors of length ℓ.

4 Conclusion

In this note, we instantiate a construction of IP-SK-ABE in the random oracle model with practical efficiency. Our construction is achieved by instantiating the framework of Attrapadung et al. with the CPRF construction of Servan-Schreiber. The scheme achieves concrete efficiency: encryption and decryption takes only 254 microseconds for vectors of length 1,000, which has potential to be applied in real-world applications where fine-grained attribute-based access policies for encryption are important.

Acknowledgments

We thank our mentor Sacha Servan-Schreiber for his continued guidance and support. We also thank the MIT PRIMES program and its organizers Slava Gerovitch and Srini Devadas for this research opportunity. Lastly, we are grateful to our PRIMES Circle mentors Lalita Devadas and Surya Mathialagan for sparking our interest in this field.

Bibliography

[AMN+18] N. Attrapadung, T. Matsuda, R. Nishimaki, S. Yamada, and T. Yamakawa. Constrained PRFs for NC^1 in traditional groups. In *CRYPTO 2018, Part II*, *LNCS* 10992, pages 543–574. Springer, Cham, August 2018.

[ILL89] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24. ACM Press, May 1989.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO'84*, *LNCS* 196, pages 47–53. Springer, Berlin, Heidelberg, August 1984.

[SS24] S. Servan-Schreiber. Constrained pseudorandom functions for inner-product predicates from weaker assumptions. In *ASIACRYPT 2024, Part II, LNCS 15485*, pages 232–265. Springer, Singapore, December 2024.

[SW05] A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In *EUROCRYPT 2005*, *LNCS* 3494, pages 457–473. Springer, Berlin, Heidelberg, May 2005.