Need for Speed: Leveraging the Power of Functional Encryption for Resource-Constrained Devices

Eugene Frimpong[1][a], Alexandros Bakas[2][b], Camille Foucault[1][c] and Antonis Michalas[1][d]

[1]*Tampere University, Tampere, Finland*

[2]*Nokia Bell Labs, Espoo, Finland*

Keywords: Elliptic Curve Cryptography, Functional Encryption, Lightweight Cryptography, Wireless Sensor Networks.

Abstract: Functional Encryption (FE) is a cutting-edge cryptographic technique that enables a user with a specific functional decryption key to determine a certain function of encrypted data without gaining access to the underlying data. Given its potential and the fact that FE is still a relatively new field, we set out to investigate how it could be applied to resource-constrained environments. This work presents what we believe to be the first lightweight FE scheme explicitly designed for resource-constrained devices. We also propose a use case protocol that demonstrates how our scheme can secure an Internet of Things (IoT) architecture where relevant devices collect data and securely deliver them to a storage server, where an analyst can request access to the encrypted data. Finally, we conduct thorough experiments on two commercially available resource-constrained devices to provide compelling evidence of our approach's practicality and efficiency. Although the results of our evaluations show that there is room for improvement in the proposed scheme, this work represents one of the first attempts to apply FE to the IoT setting that can directly impact people's daily lives and the everyday operations of organizations.

1 INTRODUCTION

The term Internet of Things (IoT) refers to a collection of varying devices or appliances utilized in different sectors that broaden the concept of the Internet. A popular IoT environment, commonly referred to as a Wireless Sensor Network (WSN), consists of a myriad of resource-constrained devices deployed in various capacities, such as sensing environmental data and monitoring human interactions. The data collected by these devices often tend to be confidential or critical, opening the network to many security and privacy-related threats that continue to draw much attention from Academia and Industry alike. For example, consider a scenario where some sensor devices have been deployed to monitor the health vitals of athletes in a particular sports competition. The data collected by these devices should be kept confidential and only available to authorized entities. However, the implementation and adoption of standard security schemes, such as public key cryptography, is complex due to the resource constraints of the devices involved (Frimpong et al., 2020). These devices typically have limited memory, computational capabilities, and power resources compared to traditional computing systems.

One of the key challenges when proposing and designing security schemes or applications for resource-constrained devices is ensuring that the application efficiently utilizes the device's resources. This often involves using techniques such as code optimization and efficient memory management. Additionally, it may be necessary to limit the application's functionality to run efficiently on the device and utilize cryptographic primitives with low memory requirements and low computational overheads. Another challenge is to design for power efficiency and low power consumption as these devices are often battery-powered. This means minimizing the device's power usage and power-saving features is critical. To this end, research in lightweight cryptography for resource-constrained devices has become a mainstay to provide privacy, trust, and integrity, focusing on reduced implementation sizes, energy costs, and computational latencies. In a recent survey of over 54 lightweight cryptographic primitives (Dhanda et al., 2020), a comparison was conducted between various lightweight ciphers with micro-controller chip energy and power consumption, hardware performance,

[a] https://orcid.org/0000-0002-4924-5258
[b] https://orcid.org/0000-0002-0731-1851
[c] https://orcid.org/0009-0001-6741-887X
[d] https://orcid.org/0000-0002-0189-3520

Frimpong, E., Bakas, A., Foucault, C. and Michalas, A.
Need for Speed: Leveraging the Power of Functional Encryption for Resource-Constrained Devices.
DOI: 10.5220/0012626200003705
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 9th International Conference on Internet of Things, Big Data and Security (IoTBDS 2024)*, pages 62-74
ISBN: 978-989-758-699-6; ISSN: 2184-4976
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

and latency. From their results, the authors observed that Elliptic Curve Cryptography (ECC) primitives were the most suitable for building lightweight cryptographic schemes. ECC schemes use an elliptic curve defined over a finite field $GF(p)$, for a large prime p, denoted by $E(GF(p))$, containing affine points $(x,y) \in GF(p) \times GF(p)$. Subsequently, the security of schemes based on ECC relies on the hardness of the Elliptic Curves Discrete Logarithm Problem (ECDLP) (Darrel Hankers and Vanstone, 2004). The primary advantage of using ECC primitives is the utilization of shorter keys, which lead to a lower memory requirement and faster field arithmetic operations (Lara-Nino et al., 2018). By relying on these advantages, we are not only able to deploy traditional ECC on constrained devices, but we take this work a step further by designing, to the best of our knowledge, the first Functional Encryption (FE) (Boneh et al., 2011) scheme that runs efficiently on resource-constrained devices.

FE is an emerging cryptographic technique that allows selective computations over encrypted data. Generally, FE schemes provide a key generation algorithm that outputs decryption keys with remarkable capabilities. More precisely, each decryption key sk_f is associated with a function f. In contrast to traditional cryptographic techniques, using sk_f on a ciphertext $\mathsf{Enc}(x)$ does *not* recover x but a function $f(x)$ – thus keeping the actual value x private. While the first constructions of FE allowed the computation of a function over a *single* ciphertext, more recent works (Goldwasser et al., 2014a) introduced the more general notion of multi-input FE (MIFE). In a MIFE scheme, given ciphertexts $\mathsf{Enc}(x_1),\ldots,\mathsf{Enc}(x_n)$, a user can use sk_f to get $f(x_1,\ldots,x_n)$. The function f allows the functional key holder to only learn highly processed data forms. Unfortunately, while MIFE seems to be a perfect fit for many real-life applications – such as cloud-based applications where multiple users store large volumes of data in remote and possibly corrupted entities – most of the works in the field revolve around constructing *generic* schemes that do not support devices with limited resources. Having identified FE as an important member of the family of modern encryption schemes that can push us into uncharted technological terrain, we make a first attempt at designing and implementing an efficient yet secure FE scheme for resource-constrained devices. This work seeks to smooth out the identified asymmetries between theory and practice while proving that resource-constrained devices can run said schemes.

Contributions. The contributions of this paper are;

C1. We first propose and design an ECC-based lightweight MIFE scheme for adding vectors' components inspired by the generic ElGamal-based FE construction presented in (Bakas et al., 2022b).

C2. Based on the proposed MIFE scheme, we construct a use-case protocol that demonstrates the applicability of our work in a real-world application along with extensive security analysis.

C3. Finally, we implement and evaluate the performance of our scheme on two commercially available resource-constrained devices and a standard desktop machine to demonstrate the proposed benefits and applicability.

2 RELATED WORK

Functional Encryption was first formalized as a generalization of public-key encryption in (Boneh et al., 2011). Since then, numerous studies with general definitions and generic constructions of FE have been proposed (Sahai and Seyalioglu, 2010; Waters, 2015; Goldwasser et al., 2013; Goldwasser et al., 2014b; Bakas et al., 2022a). Despite the promising works that have been published; there is a clear lack of works proposing FE schemes supporting use cases – a necessary step that would allow FE to be deployed in real-world applications. In particular, and to the best of our knowledge, the number of supported functionalities is currently limited to sums (Bakas et al., 2022b; Bakas et al., 2020; Bakas and Michalas, 2020), inner products (Abdalla et al., 2018; Abdalla et al., 2017; Abdalla et al., 2015) and quadratic polynomials (Sans et al., 2018). While these works are promising, unfortunately, we can not use any of them as a black box in our solution. This is because most of these works rely on cryptographic primitives over algebraic finite fields; hence, the size of the cryptographic keys becomes too large to store on a constrained device. Building on the ElGamal-based FE scheme discussed in (Bakas et al., 2022b), we've created a new lightweight FE scheme for addition. This new scheme uses an elliptic curve version of the ElGamal system. We should point out that we're not looking at the LWE-based scheme from the same source (Bakas et al., 2022b), mainly because it would require larger keys, making it a poor fit for devices with limited resources.

While the problem of aggregating encrypted data from constrained devices has been thoroughly examined in multiple works (Vinodha and Mary Anita, 2018; Castelluccia et al., 2009; Li et al., 2015;

Wang et al., 2022), we consider our construction to be the first attempt to tackle the problem using FE. For example, in (Castelluccia et al., 2009), authors propose a provably secure encryption scheme enabling additive aggregation of encrypted data. In this work, each device in a sink-rooted spanning tree environment sums all values it receives from its children, with the final value sent to a sink device. Similarly, authors in (Li et al., 2015) proposed a fully homomorphic encryption (FHE) based on a secure data aggregation scheme with simulation results. Another work (Wang et al., 2022) designed a non-interactive privacy-preserving data aggregation for resource-constrained devices with the help of a Trusted Execution Environment. Given the vast possibilities and advantages of a cryptographic primitive such as FE, we see our work as a first step towards capturing the true potential of FE in real-world solutions. To this end, we consider our approach fresh and unrelated to works relying, for example, on homomorphic encryption or proxy servers as discussed in existing works.

Finally, the efficient design and implementation of lightweight cryptographic schemes on resource-constrained devices is not new and continues to draw much attention (Lara-Nino et al., 2018; Frimpong and Michalas, 2020; Frimpong et al., 2020; He et al., 2019; Frimpong et al., 2021; Dubrova et al., 2018). In (Frimpong and Michalas, 2020), authors implement a lightweight ECC library in the Contiki-NG operating system to secure communication between multiple parties without needing a third party. Authors in (Frimpong et al., 2020) also designed and implemented a forward private dynamic Symmetric Searchable Encryption (SSE) scheme compatible with resource-constrained devices by splitting the core functionalities of the scheme between the constrained devices and a fog node. In (He et al., 2019), authors designed a lightweight certificate enrollment protocol for constrained IoT devices that utilized the Enrollment over Secure Transport (EST) semantics and leveraged the existing IoT stack to optimize resource usage. A lightweight message authentication scheme based on cyclic redundancy check (CRC) was also proposed in (Dubrova et al., 2018). Furthermore, a certificateless group key distribution protocol compatible with constrained devices was proposed in (Frimpong et al., 2021). These works achieved efficient and secure lightweight schemes that provide a foundation for a safe IoT ecosystem. We build on these foundations to design and implement our proposed lightweight FE scheme, which we prove to be efficient and secure even on the most resource-constrained devices.

3 BACKGROUND

3.1 Notation

A public/private key pair of a public-key encryption scheme PKE is denoted as $(\mathsf{pk},\mathsf{sk})$. If the public/private key pair is associated with an entity E, then we write $(\mathsf{pk}_E,\mathsf{sk}_E)$. Vectors are denoted in bold as $\mathbf{x} = (x_1,\dots,x_n)$. A PPT adversary $\mathcal{ADV}$ is a randomized algorithm for which there exists a polynomial $p(z)$, such that for all valid inputs z, the running time of $\mathcal{ADV}(z)$ is bounded by $p(|z|)$. A function $negl(\cdot)$ is called negligible if it grows slower than any inverse polynomial.

3.2 Public Key Encryption

Definition 1 (Public-Key Encryption)**.** A public-key encryption scheme PKE for a message space $\mathcal{X}$ consists of three algorithms $\mathsf{PKE} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$. A PKE scheme is said to be correct if:

$$Pr[\mathsf{Dec}(\mathsf{sk},c) \neq x \in \mathcal{X}]|[(\mathsf{pk},\mathsf{sk}) \leftarrow Gen(1^{\lambda})] \wedge [c \leftarrow \mathsf{Enc}(\mathsf{pk},x)] = negl(\lambda) \quad (1)$$

where $negl(\lambda)$ is a negligible function in the security parameter λ.

Definition 2 (Additive Ciphertext Homomorphism (ACH))**.** We say that a public-key encryption scheme PKE is additive ciphertext homomorphic if:

$$\sum_{i=1}^{n} \mathsf{Enc}(\mathsf{pk_i},\mathsf{x_i}) = \mathsf{Enc}\left(\sum_{i=1}^{n} \mathsf{pk}_i, \sum_{i=1}^{n} x_i\right) \quad (2)$$

where the first sum is the addition on the ciphertext space, the second the addition on the public key space and the third the addition on the message space.

Definition 3 (Additive Key Homomorphism (AKH))**.** We say that a public-key encryption scheme PKE is additive key homomorphic if for every two public/private key pairs $(\mathsf{pk}_1,\mathsf{sk}_1)$ and $(\mathsf{pk}_2,\mathsf{sk}_2)$ generated by $\mathsf{PKE.Gen}$, we can generate a new valid public/private key pair as $(\mathsf{pk}_1+\mathsf{pk}_2,\mathsf{sk}_1+\mathsf{sk}_2)$, where $\mathsf{pk}_1+\mathsf{pk}_2$ is the addition in the space of public keys and $\mathsf{sk}_1+\mathsf{sk}_2$ is the addition in the space of secret keys.

Definition 4 (selective-Indistinguishability-Based Security)**.** For a PKE scheme $\mathsf{PKE} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ we define the following experiments:

```
$Exp^{s-IND-CPA-\beta}(\mathcal{ADV})$

Initialize($\lambda, x_0, x_1$)                      Finalize($\beta'$)
$(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^{\lambda})$                $\beta' = \beta$
Return pk
Challenge()
$c_\beta \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_\beta)$
```

The advantage ε of $\mathcal{ADV}$ is defined as:

$$\varepsilon = \Big| Pr[Exp^{s-ind-CPA-0}(\mathcal{ADV}) = 1] \\ - Pr[Exp^{s-ind-CPA-1}(\mathcal{ADV}) = 1] \Big| \quad (3)$$

We say that PKE is s-IND-CPA-β secure if

$$\varepsilon = negl(\lambda)$$

3.3 ElGamal over Elliptic Curves

As previously stated, the proposed construction relies on the Elgamal cipher defined over elliptic curves. Hence, we begin by informally recalling how the cryptosystem works (Abdalla et al., 2015). Let E be an elliptic curve defined over $GF(p)$, where p is a large prime. The ElGamal secret key is then an element $s \in GF(p)$, while the corresponding public key is computed as $Y = s \cdot G \in E$, where G is a generator of E. To encrypt a message $x \in GF(p)$, we first pick a random $r \in [1, N-1]$, where N is the order of the curve, and in the end, we compute $c = \mathsf{Enc}(x) = (P, Q) = (r \cdot G, x \cdot G + r \cdot Y)$. Finally, for decryption, we can recover the original message using the private key s, by computing $-s \cdot P + Q$. A more formal description is illustrated in Figure 1.

ElGamal over Elliptic Curves.

Let E be an elliptic curve of order N, defined over $GF(p)$, where p is a large prime. Then, the ElGamal cryptosystem over elliptic curves is defined as a tuple $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, such that:

```
PKE.Gen(1^λ)
1. Pick a generator $G$ of $E$
2. Sample the private key as $s \xleftarrow{\$} GF(p)$
3. Compute the public key as $Y = s \cdot G$
4. Output $(s, Y)$

PKE.Enc(Y, x)
1. Sample $r \xleftarrow{\$} [1, N-1]$
2. Compute $c = (P, Q) = (r \cdot G, x \cdot G + r \cdot Y)$
3. Output c

PKE.Dec(s, c)
1. Output $-s \cdot P + Q$
```

Figure 1: ElGamal over Elliptic Curves.

3.4 From Points to Integers

While addition over an elliptic curve is possible, we can only add points on the curve, which is not very useful in the context of functional encryption. To this end, we define a mapping function as in (Ugus et al., 2009) with the purpose of mapping elements of $GF(p)$ to the curve as follows:

Definition 5 (Mapping function)**.** Let E be an elliptic curve defined over $GF(p)$, where p is a large prime. Moreover, let G be a generator of E. Then we define a mapping function $f_{map} : GF(p) \to E$ such that:

$$\forall x \in GF(p) : f_{map}(x) = x \cdot G = X \in E, \quad (4)$$

where $\cdot$ refers to point multiplication over E. For the rest of this paper, we write xG to simplify reading.

Note that when the curve is defined over a field of prime order, every element in the field is a generator of the curve. Hence, finding a generator is trivial. With the help of f_{map}, we can add elements in $GF(p)$, using points on the curve. This is possible because by definition, we have that:

$$\begin{aligned} &f_{map}(x_1) + \cdots + f_{map}(x_n) \\ &\quad = x_1G + \cdots + x_nG \\ &\quad = (x_1 + \cdots + x_n)G \\ &\quad = f_{map}(x_1 + \cdots + x_n), \end{aligned} \quad (5)$$

where each $x_i \in GF(p)$ and $f_{map}(x_i) \in E$. Note that, unambiguously, we use the same notation for addition in $GF(p)$ and its underlying addition in E. While the inverse function f_{map}^{-1} exists, computing it is equivalent to solving the DLP over an elliptic curve. However, when each x_i is small (i.e. not of cryptographic size), this is possible by relying on a list of precomputed values. Considering x_i small is realistic as data generated or collected by the devices in question will not be of cryptographic size.

3.5 Multi-Input Functional Encryption

Definition 6 (Multi-Input Functional Encryption)**.** A Multi-Input Functional Encryption scheme MIFE for a message space $\mathcal{X}$, where $\mathcal{X}$ is a vector space of dimension n, is a tuple of four algorithms $MIFE = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ such that:

- $\mathsf{Setup}(1^\lambda)$: The Setup algorithm takes a security parameter λ as input and outputs a master public/private key $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{Enc}(\mathsf{mpk}, \mathbf{x})$: The encryption algorithm is a probabilistic algorithm. Upon input the master public key mpk and a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X}$ outputs a ciphertext $\mathbf{c} = (c_1, \ldots, c_n)$.
- $\mathsf{KeyGen}(\mathsf{msk}, f)$: The key generation algorithm KeyGen is a deterministic algorithm. Upon input the master secret key msk and a function f, outputs a functional decryption key sk_f.
- $\mathsf{Dec}(\mathsf{sk}_f, \mathbf{c})$ The decryption algorithm Dec is a deterministic algorithm. Upon input a functional decryption key sk_f and a ciphertext $\mathbf{c}$, outputs $f(\mathbf{x}) = f(x_1, \ldots, x_n)$

A MIFE scheme is said to be correct if and only if:

$$Pr[\mathsf{Dec}(\mathsf{sk}_f,\mathbf{c}) \neq f(x)] | [(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^{\lambda})]$$
$$\wedge\ [\mathbf{c} \leftarrow \mathsf{Enc}(\mathsf{mpk},\mathbf{x})]$$
$$\wedge\ [\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk},\mathsf{sk}_f)] = negl(\lambda) \tag{6}$$

Definition 7 (MIFE Indistinguishability-Based Security)**.** For a MIFE scheme $\mathsf{MIFE} = (\mathsf{Setup},\mathsf{Enc},\mathsf{KeyGen},\mathsf{Dec})$, we define the following experiments:

```
Exp^{s-IND-FE-CPA-β}(𝒜𝒟𝒱)

Initialize(λ, x_0, x_1)
    mpk, msk ←$ Setup(1^λ)
    L ← ∅
    Output mpk
Key Generation(f)
    L ← L ∪ {f}
    sk_f ←$ KeyGen(msk, f)
    Output sk_f

Challenge()
    c_β ←$ Enc(mpk, x_β)
Finalize(β')
    If ∃ f ∈ L :
        f(x_0) ≠ f(x_1)
        Output ⊥
    Else
    β' = β
```

The advantage ε of $\mathcal{ADV}$ is defined as:

$$\varepsilon = \Big| Pr[Exp^{s-ind-FE-CPA-0}(\mathcal{ADV}) = 1]$$
$$- Pr[Exp^{s-ind-FE-CPA-1}(\mathcal{ADV}) = 1] \Big|$$

We say that MIFE is s-IND-FE-CPA-β secure iff $\varepsilon = negl(\lambda)$

3.6 Generic Multi-Input Functional Encryption Scheme for the Summation of a Vector's Components

We now recall the generic construction from (Bakas et al., 2022b), and in the next section, provide our lightweight instantiation from ECC.

Definition 8 (MIFE for the summation of a vector's components (MIFE_{sum}))**.** Let $\mathsf{PKE} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ be an IND-CPA secure cryptosystem that also fulfills the ACH and AKE properties. Then we define MIFE_{sum} as $\mathsf{MIFE}_{sum} = (\mathsf{Setup},\mathsf{Enc},\mathsf{KeyGen},\mathsf{Dec})$ where:

1. $\mathsf{Setup}(1^{\lambda},n)$: The Setup algorithm invokes the PKE's Gen algorithm and generates n public/private key pairs as $(\mathsf{pk}_1,\mathsf{sk}_1),\ldots,(\mathsf{pk}_n,\mathsf{sk}_n)$. It outputs a master public/private key pair as $\mathsf{mpk},\mathsf{msk}$, where $\mathsf{mpk} = (\mathsf{params},\mathsf{pk}_1,\ldots,\mathsf{pk}_n)$ and $\mathsf{msk} = (\mathsf{sk}_1,\ldots,\mathsf{sk}_n)$.
2. $\mathsf{Enc}(\mathsf{mpk},\mathbf{x})$: The Encryption algorithm Enc, takes as input the master public key mpk and a vector $\mathbf{x}$ and outputs $\mathbf{c} = \{c_1,\ldots,c_n\}$, where $c_i = \mathsf{Enc}(\mathsf{pk_i},x_i)$.
3. $\mathsf{KeyGen}(\mathsf{msk})$: The Key Generation algorithm takes as input the master secret key msk and outputs a functional key sk as $\mathsf{sk} = \sum_1^n \mathsf{sk_i}$.
4. $\mathsf{Dec}(\mathsf{sk},\mathbf{c})$: The Decryption Algorithm takes as input the functional key sk and an encrypted vector $\mathbf{c}$ and outputs $\mathsf{PKE.Dec}\left(\mathsf{sk},\sum_{i=1}^{n}\mathbf{c}\right)$.

Theorem 1. *Let* $\mathsf{PKE} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ *be an IND-CPA secure public-key encryption scheme that is both additive key and cipher homomorphic. Moreover, let* MIFE *be the generic multi-input Functional Encryption scheme for the summation of a vector's components presented in (Bakas et al., 2022b), that can be instantiated from* PKE*. Then* MIFE *is s-IND-FE-CPA secure.*

The proof of Theorem 1 is provided in section 7.

4 INSTANTIATION FROM ECC - FORMAL CONSTRUCTION

4.1 Instantiation from ECC

We introduce a construction relying on the Additive Ciphertext and Key homomorphic properties as per definitions 2 and 3. In short, an ElGamal ciphertext for a message x is given by:

$$\mathsf{Enc}(x) = (P,Q) = (rG, xG + rY), \tag{7}$$

where r is sampled randomly in $\mathbb{Z}_N^*$, N is the order of the curve, G is a generator of the curve and Y is the public key. As we see, each ciphertext consists of two points on the curve. Thus, we need to define a new operation that will allow us to add ciphertexts together:

Randomness Re-Use. In a setting involving multiple clients, ElGamal maintains its security even when the same random value r is reused, as shown in (Bellare et al., 2002). This differs from the traditional single-client ElGamal setup, where reusing the same randomness r would compromise security. However, in scenarios with $n > 1$ distinct key pairs, this vulnerability does not apply.

Definition 9 (Addition of Elgamal Ciphertexts over Elliptic Curves)**.** Assuming that we have two messages x_1 and x_2 with corresponding ciphertexts

$\mathsf{Enc}(x_1)$ and $\mathsf{Enc}(x_2)$, we define the following operation, denoted by $\oplus$:

$$\begin{aligned}\mathsf{Enc}(x_1) \oplus \mathsf{Enc}(x_2) &= (rG, x_1G + rY_1) \\ &\quad \oplus (rG, x_2G + rY_2) \\ &= (2rG, (x_1 + x_2)G + r(Y_1 + Y_2)) \\ &= (P', Q'),\end{aligned} \tag{8}$$

where the '+' symbol on the right side of the equation, denotes the standard addition and $P', Q' \in E$.

To prove that the generic MIFE construction can be instantiated from ElGamal over ECC, all we need to do is prove that the ECC variation of ElGamal is both ACH and AKH as per definitions 2 and 3. In particular, we prove the following theorem;

Theorem 2. *Let MIFE be the generic multi-input functional encryption scheme presented in (Bakas et al., 2022b). Then, it is possible to instantiate* MIFE *using ElGamal's variation over elliptic curves as the public-key encryption scheme* PKE.

Proof. As already stated, we prove that ElGamal's variation over elliptic curves is both additive key and cipher homomorphic as per definitions 2 and 3.

- **Additive Key Homomorphic.** Let $s_1, \dots, s_n \in GF(p)$ be n private keys and $Y_1, \dots Y_n$ be their respective public keys. Recall that $Y_i := s_iG, i \in [n]$. Therefore,

$$\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} s_iG = \left(\sum_{i=1}^{n} s_i\right) \cdot G$$

  Since $GF(p)$ is closed under addition, $\sum_{i=1}^{n} s_i \in GF(p)$. Hence, $\sum_{i=1}^{n} s_i$ is a valid secret key and $\sum_{i=1}^{n} Y_i$ is a valid corresponding public key by construction. Thus, the variation of ElGamal over elliptic curves is additive key homomorphic.

- **Additive Cipher Homomorphic.** Let $x_1, \dots x_n \in GF(p)$ be n measurements and $c_1, \dots c_n$ be their respective ciphertexts. Recall that $c_i := (rG, x_iG + rY), i \in [n]$, for a randomness $r \in \mathbb{Z}_N^*$ and the public key $Y \in E$. Therefore,

$$\begin{aligned}\bigoplus_{i=1}^{n} c_i &= \bigoplus_{i=1}^{n} (rG, x_iG + rY) \\ &= \left(\sum_{i=1}^{n} rG, \sum_{i=1}^{n} x_iG + \sum_{i=1}^{n} rY\right) \\ &= \left((nr) \cdot G, \sum_{i=1}^{n} x_iG + (nr) \cdot Y\right),\end{aligned}$$

  where $\oplus$ is the operation defined in equation 8. Hence, $\bigoplus_{i=1}^{n} c_i$ is a valid encryption of $\sum_{i=1}^{n} x_i$ for the same public key Y and a randomness $nr \in \mathbb{Z}_N^*$ – since $\mathbb{Z}_N^*$ is closed under addition – and thus the variation of ElGamal over elliptic curves is additive cipher homomorphic.

□

A direct result of Theorem 1 and Theorem 2, is that our instantiation is $s-IND-FE-CPA$-secure as per definition 7.

4.2 System Model

Before presenting the formal use case of the proposed construction, we first provide a description of the system model we consider. Our setup consists of four entities: *(i)* Edge Devices (**ED**), *(ii)* Data Owner (**O**), *(iii)* Storage Server (**SS**), and *(iv)* Analyst (**A**).

- **Edge Devices (ED).** Let $\mathcal{D} = \{d_1, \dots, d_n\}$ be a set of resource-constrained sensor devices deployed in an environment to collect data x. Each device encrypts collected data and sends it to the storage server.
- **Data Owner (O).** We assume the existence of a fully trusted data owner **O**, who administers and is responsible for a set of edge devices in a specific environment. **O** is responsible for generating functional decryption keys for the particular function for data collected by the devices.
- **Storage Server (SS).** We assume the existence of a storage server **SS**, an abstract external storage platform that will store encrypted data received from each device d_i. Once access to data is requested, **SS** returns the data to the requesting party.
- **Analyst (A).** Let **A** be an entity that wishes to learn the result of a specific function computed for the encrypted data collected from a set of devices $\mathcal{D}'$ such that $\mathcal{D}' \subseteq \mathcal{D}$.

5 USE CASE PROTOCOL

We now formally describe a detailed protocol to demonstrate the applicability of the proposed construction. For this use case, we consider a scenario where a device owner **O** possesses n number of sensor devices $(d_1, \dots, d_n)$. Additionally, we assume the existence of a storage server **SS** where encrypted data from the sensor nodes will be stored. A complete overview of the protocol is illustrated in Figure 2. In the proposed use case, each device d_i collects environmental or event data x_i, encrypts, and uploads the

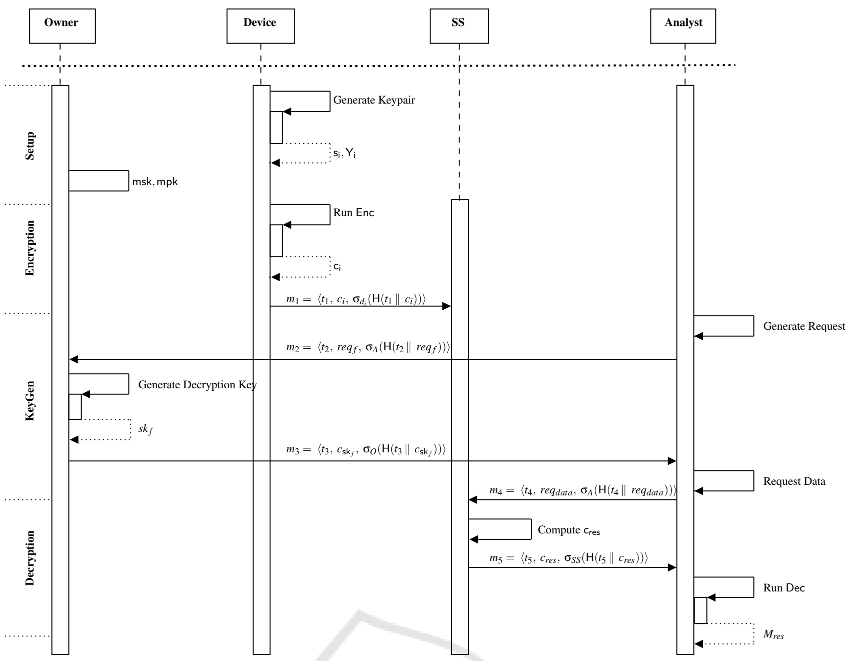
Owner
Device
SS
Analyst
Setup
Encryption
KeyGen
Decryption
Generate Keypair
s_i, Y_i
msk, mpk
Run Enc
c_i
$m_1 = \langle t_1, c_i, \sigma_{d_i}(H(t_1 \| c_i))\rangle$
Generate Request
$m_2 = \langle t_2, req_f, \sigma_A(H(t_2 \| req_f))\rangle$
Generate Decryption Key
sk_f
$m_3 = \langle t_3, c_{sk_f}, \sigma_O(H(t_3 \| c_{sk_f}))\rangle$
Request Data
$m_4 = \langle t_4, req_{data}, \sigma_A(H(t_4 \| req_{data}))\rangle$
Compute c_{res}
$m_5 = \langle t_5, c_{res}, \sigma_{SS}(H(t_5 \| c_{res}))\rangle$
Run Dec
M_{res}

Figure 2: Use Case Protocol Overview.

data to **SS**. The main building blocks of our protocol are as follows:

- An IND-CCA2 secure public key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$;
- An EUF-CMA secure signature scheme $\mathsf{S} = (sign, verify)$;
- A first and second pre-image resistant cryptographic hash function H;
- A synchronized clock between all entities.

As a first step, each device d_i runs the $\mathsf{MIFE.Setup}$ algorithm to generate a private/public key pair (s_i, Y_i) for ElGamal's cryptosystem. On completion, **O** computes msk and mpk as demonstrated in algorithm 1[1].

Algorithm 1: $\mathsf{Setup}(1^\lambda, n)$.

```
    Device $d_i$:
 1: Generate $n$ private keys as $s_i \overset{\$}{\leftarrow} GF(p)$
 2: For each $s_i$, generate the corresponding public key as $Y_i = s_i G \in E$
    Device Owner O:
 3: Output msk as $\mathsf{msk} = (s_1, \ldots, s_n)$
 4: Output mpk as $\mathsf{mpk} = (Y_1, \ldots, Y_n)$
```

Subsequently, we assume that each device d_i generates some data x_i, e.g. generated data about an event.

[1]Note: we assume the data owner has secure access to the key pair generated by each device in its ownership.

Upon generation, d_i computes $f_{map}(x_i)$, samples a random r and encrypts it as shown in line 5 of Algorithm 2 to get c_i. Finally, the encrypted measurement is outsourced to **SS** via:

$$m_1 = \langle t_1, c_i, \sigma_{d_i}(H(t_1 \| c_i))\rangle, \quad (9)$$

where t_1 is a timestamp and σ_{d_i} is d_i's signature. Upon reception, **SS** verifies the freshness and the integrity of the message m_1 by checking the timestamp and the signature, respectively. **SS** outputs $\perp$ and aborts the protocol if any verification fails. Otherwise, it will store all c_i.

Algorithm 2: $\mathsf{Enc}(\mathsf{mpk}, \mathbf{c} = (c_1, \ldots, c_n))$.

```
 1: Device $d_i$:
 2: Generate a measurement $m_i \in GF(p)$
 3: Compute $f_{map}(m_i) = m_i G = M_i \in E$
 4: Pick a random $r \in \mathbb{Z}_N^*$ , where $N$ is the order of the curve
 5: $c_i = \mathsf{Enc}(m_i) = (P, Q) = (rG, M_i + rY_i)$
 6: Send $m_1 = \langle t_1, c_i, \sigma_{d_i}(H(t_1 \| c_i))\rangle$ to SS
 7: SS:
 8: Verify the freshness and the integrity of $m_1$
 9: If any verification fails, output $\perp$ and abort the protocol
10: Store $c_i$
```

On completion of the encryption phase, we assume that an analyst **A** wishes to compute a function on encrypted data collected by $\mathcal{D}'$ stored on **SS**. To do so, **A** first requests permission from **O** to do so by sending a request of the following form:

$$m_2 = \langle t_2, \texttt{req(f)}, \sigma_A(H(t_2 \| \texttt{req(f)}))\rangle, \quad (10)$$

where `req(f)` denotes a request for the specific function f. Upon reception, **O** verifies the freshness and integrity of the message. If any verification fails. **O** outputs $\perp$ and aborts the protocol. Otherwise, **O** first computes the functional decryption key sk_f for the specific function by running the MIFE.KeyGen algorithm as described in Algorithm 3. As a next step, **O** encrypts sk_f using the public key of the analyst **A** to get a ciphertext c_{sk_f}, and finally sends it to **A** via the following message:

$$m_3 = \langle t_3, c_{\mathsf{sk}_f}, \sigma_O(H(t_3 \| c_{\mathsf{sk}_f}))\rangle \quad (11)$$

Algorithm 3: KeyGen(msk).

```
1: Analyst A:
2: Send a request for a functional decryption key, for
   a function $f$, to the data owner via $m_2$
3: Data Owner O:
4: Verify the freshness and the integrity of $m_2$
5: If any verification fails, output $\perp$ and abort the
   protocol.
6: Otherwise, compute the functional decryption
   key $\mathsf{sk}_f$ as $\mathsf{sk}_f = \sum_1^n s_i$
7: Send $\mathsf{sk}_f$ to $A$ via $m_3$
```

Finally, and upon successful completion of the previous algorithm, **A** holds the functional decryption key sk_f. At this point, **A** sends a request for the encrypted data to the **SS** via:

$$m_4 = \langle t_4, \texttt{req(data)}, \sigma_A(H(t_4 \| \texttt{req(data)}))\rangle \quad (12)$$

where `req(data)` denotes a request for encrypted data collected by the set of devices $\mathcal{D}'$. Upon reception, **SS** will verify the message's integrity and freshness. **SS** will output $\perp$ and abort the protocol if the verification fails. Otherwise, **SS** will compute $c_{res} = \bigoplus_{i=1}^n c_i = (P_{res}, Q_{res})$ and reply to **A** with the following message:

$$m_5 = \langle t_5, c_{res}, \sigma_{SS}(H(t_5 \| c_{res}))\,. \quad (13)$$

Upon reception, **A** will verify the freshness and integrity of the message. If the verification fails, **A** will output $\perp$ and abort the protocol. Otherwise, **A** will run the functional decryption algorithm as shown in algorithm 4 using sk_f to recover $f(x_1, \dots, x_n) = \sum_1^n x_i$.

Possible Extension. Our protocol can be enhanced to also facilitate the verification of the decryption process. This feature is crucial as it paves the way for addressing stronger threat models. In particular, if we assume a malicious adversary that colludes with SS,

Algorithm 4: $\mathsf{Dec}(\mathsf{sk}_f, c_1, \dots, c_n)$.

```
   Analyst A:
1: Send a request for the encrypted data to SS via
   $m_4$
   Storage Server SS:
2: Verify the freshness and the integrity of $m_4$
3: If any verification fails, output $\perp$ and abort the
   protocol
4: Otherwise compute the sum of the ciphertexts as
   $c_{res} = \bigoplus_{i=1}^n c_i = (P_{res}, Q_{res})$, where $P_{res}, Q_{res} \in E$
5: Send $c_{res}$ to $A$ via $m_5$
   Analyst A:
6: Verify the freshness and the integrity of $m_5$
7: If any verification fails, output $\perp$ and abort the
   protocol
8: Otherwise Compute $n^{-1}$ the modular inverse of $n$
   in $GF(p)$
9: Compute $-\left(n^{-1}\mathsf{sk}_f\right) \cdot P_{res} + Q_{res}$ to get $X_{res} \in E$
10: Run $f_{map}^{-1}(X_{res}) = x_1 + \cdots + x_n \in GF(p)$
```

there's a potential risk of releasing manipulated statistics to deceive data analysts. We, therefore, argue about the importance of an analyst, possessing only $\{Y_i\}_{i\in[1,n]}$, c_{res}, and X_{res}, confirming that $f_{\text{map}}^{-1}(X_{\text{res}})$ matches $\sum_{i=1}^n x_i$, without additional information disclosures. Note that this verification can be performed in a zero-knowledge fashion without access to the master secret key or the functional decryption key corresponding to the function. More specifically, it relies on a zero-knowledge proof related to a discrete logarithm over an elliptic curve. However, the topic of conducting zero-knowledge proofs on devices with limited resources is not studied in this work and is left as future work.

6 SECURITY ANALYSIS AND THREAT MODEL

In this section, we define a threat model as a list of attacks on our protocol and prove that our protocol is secure against these attacks. The security analysis of the construction relies on Theorem 1 and is proven in section 7. Additionally, the detailed security analysis of the protocol is provided in subsection 6.1. Now, to evaluate the security of our construction, we consider an active adversary $\mathcal{ADV}$ as a probabilistic polynomial time (PPT) algorithm and estimate its capabilities by a set of attacks she might try to launch. We assume that $\mathcal{ADV}$ can interact with the messages $m_1, \dots, m_5$ and model her capabilities by the following attacks.

Attack 1 (Substitution Attack)**.** Let $\mathcal{ADV}$ be a PPT adversary. $\mathcal{ADV}$ successfully performs a Substitution Attack on a message m_i if she manages to replace m_i by $m_i^{\mathcal{ADV}}$ in an indistinguishable way.

Attack 2 (Replay Attack)**.** Let $\mathcal{ADV}$ be a PPT adversary. $\mathcal{ADV}$ successfully launches a Replay Attack if she can send an outdated message m_i to a valid recipient and convinces it that this message is fresh.

6.1 Security Analysis

Based on the threat model defined in section 6, we consider an active adversary $\mathcal{A}$ who can interact at each protocol step through the messages $m_1, \dots m_5$. To demonstrate the security of our protocol, we prove the following theorem:

Theorem 3. *Let* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an IND-CCA2 secure public key encryption scheme and* $S = (sign, verify)$ *be an EUF-CMA secure signature scheme. Let H be a first and second pre-image resistant cryptographic hash function, and assume that there exists a synchronized clock between all entities. Under these assumptions, our protocol defined in section 5 is secure against an active PPT adversary* $\mathcal{A}$ *for the threat model section 6.*

Proof. We successively consider the attacks defined earlier in our threat model.

Substitution Attack Soundness. Let $\mathcal{S}$ and $\mathcal{R}$ be two distinct parties in the set $\{$**ED**, **O**, **SS**, **A**$\}$ and $m_1, \dots, m_5$ be the messages sent during the protocol. Remark that a message m_i sent from $\mathcal{S}$ to $\mathcal{R}$ is of the form $m_i = \langle t_i, x, \sigma_{\mathcal{S}}(H(t_i \| x)) \rangle$, where x is either a ciphertext or a request. To successfully perform a Substitution Attack, $\mathcal{ADV}$ intercepts m_i sent by $\mathcal{S}$ to $\mathcal{R}$ and replace x by $x^{\mathcal{ADV}}$. Assume that $\mathcal{ADV}$ is able to generate a valid value for $x^{\mathcal{ADV}}$. To replace $\sigma_{\mathcal{S}}(H(t_i \| x))$ by $\sigma_{\mathcal{S}}(H(t_i \| x^{\mathcal{ADV}}))$, $\mathcal{ADV}$ has to forge the signature $\sigma_{\mathcal{S}}$. Since S is EUF-CMA, this can only happen with a negligible probability. Thus, if λ is the security parameter for S, the adversary's advantage for this attack is $\varepsilon_1 = negl(\lambda)$.

The proof for the security against replay attack is similar since it also relies on the unforgeability of the signature.

Replay Attack Soundness. Let $\mathcal{S}$ and $\mathcal{R}$ be two distinct parties in the set $\{$**ED**, **O**, **SS**, **A**$\}$ and $m_1, \dots, m_5$ be the messages sent during the protocol. Remark that a message m_i sent from $\mathcal{S}$ to $\mathcal{R}$ is of the form $m_i = \langle t_i, x, \sigma_{\mathcal{S}}(H(t_i \| x)) \rangle$, where x is either a ciphertext or a request. To successfully perform a Replay Attack, $\mathcal{ADV}$ eavesdrops on a message m_i sent by $\mathcal{S}$ to $\mathcal{R}$, store it, and send it again to $\mathcal{R}$ later without him noticing that the message is outdated. To do so, she needs to convince $\mathcal{R}$ that m_i is fresh by updating the timestamp t_i to t_j, $i \neq j$. However, to replace t_i by t_j in $\sigma_{\mathcal{S}}(H(t_i \| x))$, $\mathcal{ADV}$ has to forge the signature $\sigma_{\mathcal{S}}$. Since S is EUF-CMA, this can only happen with a negligible probability. Thus, if λ is the security parameter for S, the adversary's advantage for this attack is $\varepsilon_2 = negl(\lambda)$.

Since the finite sum of negligible functions is also a negligible function, the overall advantage of an adversary $\mathcal{ADV}$ on this protocol is $\varepsilon = negl(\lambda)$. $\square$

Now we provide the proof of the Theorem 1. Remark that this proof is already provided in (Bakas et al., 2022b) for the general construction; hence, we follow its outline and address the specific case of addition.

7 PROOF OF THEOREM 1

Proof. Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme and $\mathcal{A}$ and $\mathcal{B}$ be two independent PPT algorithms. We set game 1 as the game for PKE shown in Definition 4, in which $\mathcal{B}$ is the adversary for a challenger $\mathcal{C}$. Besides, we set game 2 as the game for MIFE shown in Definition 7, in which $\mathcal{A}$ is the adversary and $\mathcal{B}$ is the challenger.

Setup of Game 1. The adversary $\mathcal{B}$ randomly samples an element x in the message space of PKE and sends it to the challenger $\mathcal{C}$. The latter generates a pair of keys $(pk_C, sk_C) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$, randomly picks $b \hookleftarrow \{0,1\}$ and sends back both pk_C and $c_b \leftarrow \mathsf{PKE.Enc}(pk, b \cdot x)$ to $\mathcal{B}$.

Generation of Keys for Game 2. First of all, $\mathcal{B}$ calls $\mathcal{A}$, who provides two messages $\mathbf{x_0}$ and $\mathbf{x_1}$ such that:

$$\sum_{i=1}^{n} x_{0,i} = \sum_{i=1}^{n} x_{1,i} \tag{14}$$

$\mathcal{B}$ generates a basis for a vector space $\mathcal{V} \in \mathcal{M}$ such that $\forall \mathbf{x} \in \mathcal{V}$, $\sum_{i=1}^{n} x_i = 0$ in order to ensure that every decryption key query will be made for vectors $\mathbf{x_0}, \mathbf{x_1}$ such that $\sum_{i=1}^{n} x_{1,i} = \sum_{i=1}^{n} x_{0,i}$. She samples $n-1$ independent vectors $\mathbf{r_1}, \dots, \mathbf{r_{n-1}}$, such that $\forall i \in [n-1]$, $\mathbf{r_i}$ is independent of $\mathbf{x_1} - \mathbf{x_0}$ and sets the basis to $(\mathbf{x_1} - \mathbf{x_0}, \mathbf{r_1}, \dots, \mathbf{r_{n-1}})$. $\forall i \in [n]$, set

$\alpha_i = \frac{x_{1,i} - x_{0,i}}{\|\mathbf{x_1} - \mathbf{x_0}\|_2^2}$. This construction induces the canonical vectors $\mathbf{e_i} = \left[\alpha_i \cdot (\mathbf{x_1} - \mathbf{x_0}) + \sum_1^{n-1} \mathbf{r_j}\right]$. Finally, $\mathcal{B}$ produces $(\mathsf{pk}_{\alpha_j}, \mathsf{sk}_{\alpha_j}) \leftarrow \mathsf{PKE.Gen}(1^\lambda), \forall j \in [1, n-1]$ and sets:

$$\mathsf{pk_i} = \alpha_i \cdot \mathsf{pk}_C + \sum_{j=1}^{n-1} \mathsf{pk}_{\alpha_j} \quad \text{and} \quad \mathsf{mpk} = (\mathsf{pk_1}, \ldots, \mathsf{pk_n}).$$

Remark that the associated secret keys $\mathsf{sk}_i = \alpha_i \cdot \mathsf{sk}_C + \sum_1^{n-1} \mathsf{sk}_{\alpha_j}$ are unknown from $\mathcal{B}$, who does not know sk_C. In the end, the challenger $\mathcal{B}$ sets the functional decryption key to $\mathsf{sk} = \sum_{j=1}^{n-1} \mathsf{sk}_{\alpha_j}$ using the previous construction.

Challenges for Game 2. The adversary $\mathcal{A}$ sends $\mathbf{x_0}$ and $\mathbf{x_1}$ such that $\sum_{i=1}^n x_{0,i} = \sum_{i=1}^n x_{1,i}$. $\mathcal{B}$ samples at random $\beta \hookleftarrow \{0,1\}$. In theory, she should pick $\mathbf{x}_\beta$ and send back its encryption. However, in order to break the game 1, she instead computes:

$$c = \alpha \cdot c_b + \mathsf{PKE.Enc}\left(\sum_{j=1}^{n-1} \mathsf{pk}_{\alpha_j}, 0\right) + \mathsf{PKE.Enc}(0, \mathbf{x}_\beta) \tag{15}$$

and sends c to $\mathcal{A}$. Finally, the latter outputs a guess for β. If her guess is correct, then $\mathcal{B}$ guesses that C encrypted 0. If not, $\mathcal{B}$ guesses that C encrypted x. We treat those cases separately.

If C encrypted 0, then equation 15 becomes:

$$\begin{aligned} c &= \mathsf{PKE.Enc}(\alpha \cdot \mathsf{pk}_C, 0) \\ &\quad + \mathsf{PKE.Enc}\left(\sum_{j=1}^{n-1} \mathsf{pk}_{\alpha_j}, 0\right) + \mathsf{PKE.Enc}(0, \mathbf{x}_\beta) \\ &= \mathsf{PKE.Enc}(\alpha \cdot \mathsf{pk}_C + \sum_{j=1}^{n-1} \mathsf{pk}_{\alpha_j} + 0, 0 + 0 + \mathbf{x}_\beta) \\ &= \mathsf{PKE.Enc}(\mathsf{pk}_i, \mathbf{x}_\beta) \end{aligned}$$

In this case, $\mathcal{B}$ simulates a perfect view of the environment for $\mathcal{A}$. Therefore, if $\mathcal{A}$ guesses β in game 2 with an advantage $\varepsilon_\mathcal{A}$, $\mathcal{B}$, guesses b in game 1 with an advantage $\varepsilon_\mathcal{B}$. Finally, $\varepsilon_\mathcal{A} = \varepsilon_\mathcal{B}$.

If C encrypted x, then equation 15 becomes:

$$\begin{aligned} c &= \mathsf{PKE.Enc}(\alpha \cdot \mathsf{pk}_C, \alpha \cdot x) \\ &\quad + \mathsf{PKE.Enc}\left(\sum_{j=1}^{n-1} \mathsf{pk}_{\alpha_j}, 0\right) + \mathsf{PKE.Enc}(0, \mathbf{x}_\beta) \\ &= \mathsf{PKE.Enc}(\alpha \cdot \mathsf{pk}_C + \sum_{j=1}^{n-1} \mathsf{pk}_{\alpha_j} + 0, \alpha \cdot x + 0 + \mathbf{x}_\beta) \\ &= \mathsf{PKE.Enc}(\mathsf{pk}_i, \alpha \cdot x + \mathbf{x}_\beta) \end{aligned}$$

Since $\alpha = \frac{\mathbf{x}_1 - \mathbf{x}_0}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2^2}$, we have

$$\begin{aligned} \alpha \cdot x + \mathbf{x}_\beta &= \frac{x}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2^2}(\mathbf{x}_1 - \mathbf{x}_0) + \mathbf{x}_\beta \\ &= \frac{x}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2^2}(\mathbf{x}_1 - \mathbf{x}_0) + \mathbf{x}_0 + \beta(\mathbf{x}_1 - \mathbf{x}_0) \end{aligned}$$

If we set $\mu := \frac{x}{\|\mathbf{x}_1 - \mathbf{x}_0\|_2^2} + \beta$, we obtain $c = \mathsf{PKE.Enc}(\mathsf{pk}_i, \mu \cdot \mathbf{x}_1 + (1 - \mu) \cdot \mathbf{x}_0)$, which corresponds to the challenge ciphertext. Remark that $\mu \cdot \mathbf{x}_1 + (1 - \mu)\mathbf{x}_0$ is well defined, as a linear combination of elements of the vector space V with unitary norm. Finally, the value of β being hidden, the advantage $\varepsilon_\mathcal{B}$ of $\mathcal{B}$ is negligible.

Conclusion. The overall advantage of $\mathcal{B}$ on game 1 is thus $\varepsilon_\mathcal{B} = \varepsilon_\mathcal{A}$, which means that the adversary $\mathcal{A}$ can break our MIFE protocol if $\mathcal{B}$ is able to break the PKE construction. By contraposition, since PKE is IND-CPA secure, our construction is also IND-CPA secure, which concludes. □

8 EXPERIMENTAL EVALUATION

This section provides an evaluation of our proposed construction's performance and energy efficiency on two commercially available resource-constrained devices. To this end, our test bed consisted of the two sensor devices: an Arm Cortex-M4 nrf52840dk board[2], and an ARM Cortex-M3 Zolertia Re-Mote board[3]. For the rest of this section, we refer to the nrf52840dk board as ***nrf*** and the Zolertia Re-Mote board simply as ***zolertia***. Additionally, to provide a more comprehensive evaluation of our implementation, we measured its performance on a standard Desktop – referred to as ***standard***.

Experiment Testbed

- **zolertia:** A 32 MHz ARM Cortex-M3 Zolertai Re-Mote RevB board. Configuration: [512KB Flash, 32KB RAM].
- **nrf:** A 64 MHz ARM Cortex-M4 nrf52840dk board. Configuration: [1MB Flash, 32KB RAM]
- **standard:** An Intel Core i7-4790 desktop running Ubuntu 20.04 LTS. Configuration: [8 CPUs, 16GB RAM].

To test the necessary functions, we developed a Contiki-NG (Oikonomou et al., 2022) application using the c25519[4] cryptographic library for all cryptographic operations. Due to the resource constraints

[2]https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dk

[3]https://docs.contiki-ng.org/en/develop/doc/platforms/zolertia/Zolertia-RE-Mote-platform-(revision-B).html

[4]https://www.dlbeer.co.nz/oss/c25519.html

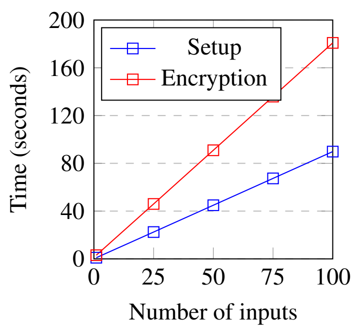

Figure 3: nrf.

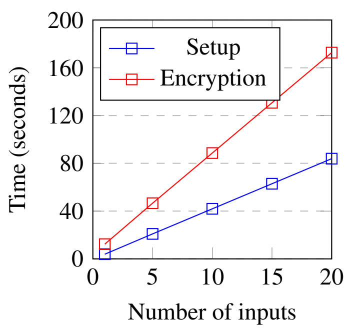

Figure 4: Zolertia.

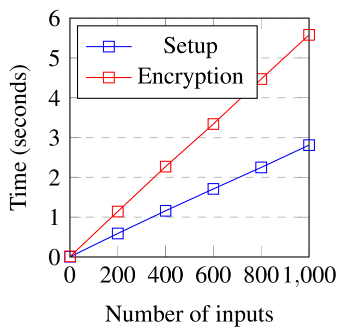

Figure 5: Standard.

of the devices involved in these experiments, we focused primarily on analyzing the execution times and energy consumption of each algorithm. Also, for each device, we implemented all the algorithms introduced in section 5, i.e. instead of limiting our evaluations to only algorithms that a resource-constrained device would ideally execute, we executed all algorithms on each device to demonstrate the overall efficiency of our work. All programming for this section was written in C, and each experiment was conducted 50 times with the average results taken.

8.1 Performance of Cryptographic Algorithms

To measure the performance of our work, we primarily focused on the computational performance of each algorithm. More precisely, we measured each device's time to execute a specific algorithm successfully. In Contiki-NG, the system time is measured in CPU ticks and is represented as long unsigned values. As a result of this implementation feature, all performance values observed from our experiments were recorded in ticks and converted to seconds. The specific figures were derived by dividing the number of recorded ticks by 128 (CPU ticks per second is 128, as defined in Contiki-NG). For these experiments, we resorted to the use case protocol described in section 5, which we used as the basis of our evaluations. However, we did not measure the computational costs of the public key encryption and secure signature schemes used to provide confidentiality and authentication of the exchanged messages or any communication costs involved.

First, we assume that each device executes the Setup and Encrypt ion algorithms. Subsequently, we consider a scenario where a device, upon receiving n number of ciphertexts, where $n \in [2, 100]$ executes the Keygen algorithm, and finally, the Decryption algorithm to retrieve the results in plain. Overall, it took the nrf board 0.97 seconds to run Setup, 3.07 seconds to run Encryption, and 1.22 seconds to run the Decryption for 2 ciphertexts. For the zolertia board, the execution times were 3.9 seconds for Setup, 12.3 seconds for Encryption, and 5.41 seconds for Decryption for 2 ciphertexts. Finally, on the standard machine, the execution times were 0.005 seconds for Setup, 0.015 seconds for Encryption, and 0.006 seconds to run the Decryption algorithm for 2 ciphertexts (Table 1). The KeyGen algorithm was executed for all devices with negligible computational overhead (e.g., for both zolertia and nrf boards, it registered 0 ticks). These results show that the cost of generating the functional decryption key using the KeyGen algorithm did not incur any computational overhead. Additionally, the most expensive algorithm for all devices is the Encryption algorithm, which involves three ECC point multiplications (Algorithm 2).

Table 1: Function Execution Times.

	nrf		zolertia		standard
Functions	**CPU Ticks**	**Time(sec)**	**CPU Ticks**	**Time(sec)**	**Time(sec)**
Setup	125	0.97	506	3.9	0.005
Encryption	393	3.07	1577	12.3	0.015
KeyGen	0	0	0	0	0.000
Decryption	156	1.22	693	5.41	0.006

As a next step, we measured the cost of executing the Setup and Encryption algorithms for a varying number of inputs from 1 to 100 on the nrf board (Figure 3), 1 to 20 on the zolertia board (Figure 4), and 1 to 1000 on the standard machine (Figure 5). Experiments on the zolertia board were limited to 20 due to memory constraints; the board froze when we tried to allocate memory for more than 20 key pairs and ciphertexts. To generate 100 unique key pairs by executing the Setup algorithm, it took the nrf 89.82 seconds, while it took the zolertia board 83.9 seconds to generate 20 unique key pairs. On the other hand, the standard device required only 2.81 seconds to generate 1000 unique key pairs. Meanwhile, the nrf board encrypted 100 inputs in 180.84 seconds, the zolertia board encrypted 20 data inputs in 172.7 seconds, and the standard device encrypted 1000 data inputs in 5.58

seconds. The cost of executing the Encryption algorithm continued to be the most expensive operation compared to the Setup algorithm. In our use case scenario, we note that the device is only responsible for running the Encryption algorithm.

Finally, we measured the execution times of the KeyGen and Decryption functions for n number of ciphertexts, where $n \in [2, 100]$. Once again, due to resource constraints of the zolertia board, we could only run both functions for 20 ciphertexts. From our experiments, we observed that as the number of ciphertexts increased, there were negligible increases in the execution time of the Decrypt ion function. For example, the nrf board executed the Decryption function for two ciphertexts in 1.22 seconds and 1.39 seconds for 100 ciphertexts. Likewise, the standard machine executed the Decryption function in 0.006 seconds for two ciphertexts and 0.014 seconds for 1000 ciphertexts. The results from this section prove that our proposed construction is practical and efficient when deployed on devices with very limited resources, much like the devices used in our experiments.

Table 2: Energy Consumption.

	nrf		zolertia	
Functions	**Time (s)**	**Energy (mJ)**	**Time (s)**	**Energy (mJ)**
Setup	0.97	18.2	3.9	468
Encryption	3.07	57.7	12.3	1476
KeyGen	0	0	0	0
Decryption	1.22	22.9	5.41	649.2

8.2 Energy Consumption

In this phase of our evaluations, we utilized the Energest module provided by Contiki-NG to measure the energy consumption. The Energest module provides lightweight software-based energy estimations for resource-constrained devices by monitoring the operations of the device's various hardware components (i.e., CPU and Radio). The active modes during our evaluations were the CPU active and radio listening. We observed that the energy measurements correlate directly to the execution times of the underlying algorithms (i.e., the longer it takes to run an algorithm, the more energy is consumed). To compute the energy values, we utilized current consumption values for CPU active and radio listening modes of the chosen boards. Finally, we only focused on the nrf and zolertia boards for these experiments as these have constrained resources, and energy consumption is essential. The nrf board used 18.2 mJ to run Setup, 57.7 mJ to run Encryption, and 22.9 mJ to execute the Decryption for 2 ciphertexts. While the zolertia board consumed 468 mJ for Setup, 1.476 J for Encryption, and 649.2 mJ to execute Decryption for 2 ciphertexts (Table 2). Once again, from these results, we observe that our construction is energy efficient.

Finally, we acknowledge that our experiments would have been more comprehensive if we compared our evaluations with other implementations; however, it is worth noting that we found no FE schemes (including the construction in (Bakas et al., 2022b)) capable of being implemented on our sensor devices due to key size and memory constraints (one of the motivating factors for this work).

Open Science and Reproducible Research. To support open science and reproducible research, the source codes used for our evaluations have been made available online[5].

9 CONCLUSION

As the use of resource-constrained devices in vital settings becomes widespread, the need for secure solutions continues to grow. In this work, we proposed and implemented an efficient and lightweight Functional Encryption scheme using ECC primitives. Our scheme is capable of running on resource-constrained devices without causing excessive energy consumption or computational overhead. ECC primitives were chosen because of their low memory requirements and smaller key sizes. Our goal was to pave the way for the future of resource-constrained devices to rely less on traditional cryptography and instead leverage the power and versatility of modern cryptographic methods such as FE. By doing so, we aim to contribute to the ongoing effort to incorporate security measures into the Internet of Things ecosystem without compromising its core functionality.

ACKNOWLEDGEMENTS

This work was funded by the HARPOCRATES (No 101069535) and FACILITATE (No 101034366) projects, Horizon Europe

REFERENCES

Abdalla, M., , D., Fiore, D., Gay, R., and Ursu, B. (2018). Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *Advances in Cryptology – CRYPTO 2018*.

[5]https://github.com/iammrgenie/sumFE_v2

Abdalla, M., Bourse, F., De Caro, A., and Pointcheval, D. (2015). Simple functional encryption schemes for inner products. In *IACR International Workshop on Public Key Cryptography*, pages 733–751. Springer.

Abdalla, M., Gay, R., Raykova, M., and Wee, H. (2017). Multi-input inner-product functional encryption from pairings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer.

Bakas, A. and Michalas, A. (2020). Multi-input functional encryption: efficient applications from symmetric primitives. In *IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1105–1112.

Bakas, A., Michalas, A., and Dimitriou, T. (2022a). Private lives matter: A differential private functional encryption scheme. In *Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy*, CODASPY '22, page 300–311, New York, NY, USA. Association for Computing Machinery.

Bakas, A., Michalas, A., Frimpong, E., and Rabaninejad, R. (2022b). Feel the quantum functioning: Instantiating generic multi-input functional encryption from learning with errors. In *Data and Applications Security and Privacy XXXVI: 36th Annual IFIP WG 11.3 Conference, DBSec 2022*, page 279–299. Springer-Verlag.

Bakas, A., Michalas, A., and Ullah, A. (2020). (f) unctional sifting: A privacy-preserving reputation system through multi-input functional encryption. In *Nordic Conference on Secure IT Systems*, pages 111–126.

Bellare, M., Boldyreva, A., and Staddon, J. (2002). Randomness re-use in multi-recipient encryption schemeas. In Desmedt, Y. G., editor, *Public Key Cryptography — PKC 2003*, pages 85–99, Berlin, Heidelberg. Springer Berlin Heidelberg.

Boneh, D., Sahai, A., and Waters, B. (2011). Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer.

Castelluccia, C., Chan, A. C.-F., Mykletun, E., and Tsudik, G. (2009). Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Transactions on Sensor Networks*, 5(3):1–36.

Darrel Hankers, S. and Vanstone, A. M. (2004). Guide to elliptic curve cryptography. *Springer Professional Computing*, pages XX–312.

Dhanda, S. S., Singh, B., and Jindal, P. (2020). Lightweight cryptography: A solution to secure iot. *Wireless Personal Communications*, 112(3):1947–1980.

Dubrova, E., Näslund, M., Selander, G., and Lindqvist, F. (2018). Lightweight message authentication for constrained devices. *Proceedings of the 11th ACM Conference on Security and Privacy in Wireless and Mobile Networks*.

Frimpong, E., Bakas, A., Dang, H.-V., and Michalas, A. (2020). Do not tell me what i cannot do! (the constrained device shouted under the cover of the fog): Implementing symmetric searchable encryption on constrained devices. *Proceedings of the 5th International Conference on Internet of Things, Big Data and Security*.

Frimpong, E. and Michalas, A. (2020). Secon-ng: Implementing a lightweight cryptographic library based on ecdh and ecdsa for the development of secure and privacy-preserving protocols in contiki-ng. *Proceedings of the 35th Annual ACM Symposium on Applied Computing*.

Frimpong, E., Rabbaninejad, R., and Michalas, A. (2021). Arrows in a quiver: A secure certificateless group key distribution protocol for drones. *Secure IT Systems*, page 31–48.

Goldwasser, S., Gordon, S. D., Goyal, V., Jain, A., Katz, J., Liu, F.-H., Sahai, A., Shi, E., and Zhou, H.-S. (2014a). Multi-input functional encryption. In Nguyen, P. Q. and Oswald, E., editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 578–602, Berlin, Heidelberg. Springer Berlin Heidelberg.

Goldwasser, S., Gordon, S. D., Goyal, V., Jain, A., Katz, J., Liu, F.-H., Sahai, A., Shi, E., and Zhou, H.-S. (2014b). Multi-input functional encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 578–602. Springer.

Goldwasser, S., Kalai, Y. T., Popa, R. A., Vaikuntanathan, V., and Zeldovich, N. (2013). How to run turing machines on encrypted data. In *Annual Cryptology Conference*, pages 536–553. Springer.

He, Z., Furuhed, M., and Raza, S. (2019). Indraj: digital certificate enrollment for battery-powered wireless devices. *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*.

Lara-Nino, C. A., Diaz-Perez, A., and Morales-Sandoval, M. (2018). Elliptic curve lightweight cryptography: A survey. *IEEE Access*, 6:72514–72550.

Li, X., Chen, D., Li, C., and Wang, L. (2015). Secure data aggregation with fully homomorphic encryption in large-scale wireless sensor networks. *Sensors*, 15(7):15952–15973.

Oikonomou, G., Duquennoy, S., Elsts, A., Eriksson, J., Tanaka, Y., and Tsiftes, N. (2022). The Contiki-NG open source operating system for next generation IoT devices. *SoftwareX*, 18:101089.

Sahai, A. and Seyalioglu, H. (2010). Worry-free encryption: functional encryption with public keys. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 463–472.

Sans, E. D., Gay, R., and Pointcheval, D. (2018). Reading in the dark: Classifying encrypted digits with functional encryption. *IACR Cryptology ePrint Archive*, 2018:206.

Ugus, O., Westhoff, D., Laue, R., Shoufan, A., and Huss, S. A. (2009). Optimized implementation of elliptic curve based additive homomorphic encryption for wireless sensor networks. *CoRR*, abs/0903.3900.

Vinodha, D. and Mary Anita, E. A. (2018). Secure data aggregation techniques for wireless sensor networks: A review. *Archives of Computational Methods in Engineering*, 26(4):1007–1027.

Wang, M., He, K., Chen, J., Du, R., Zhang, B., and Li, Z. (2022). Panda: Lightweight non-interactive privacy-preserving data aggregation for constrained devices. *Future Generation Computer Systems*, 131:28–42.

Waters, B. (2015). A punctured programming approach to adaptively secure functional encryption. In *Annual Cryptology Conference*, pages 678–697. Springer.