Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 4087816, 17 pages
https://doi.org/10.1155/2022/4087816

Research Article

A Study on Scalar Multiplication Parallel Processing for X25519 Decryption of 5G Core Network SIDF Function for mMTC IoT Environment

Changuk Jang[1], **Juhong Han**[1], **Akshita Maradapu Vera Venkata Sai**[2], **Yingshu Li**[2], **and Okyeon Yi**[3]

[1]*Department of Financial Information Security at Kookmin University, Seoul 02707, Republic of Korea*
[2]*Department of Computer Science at Georgia State University, Atlanta, GA 30303, USA*
[3]*Department of Information Security Cryptology and Mathematics at Kookmin University, Seoul 02707, Republic of Korea*

Correspondence should be addressed to Okyeon Yi; oyyi@kookmin.ac.kr

Received 10 March 2022; Accepted 30 April 2022; Published 6 June 2022

Academic Editor: Qiang Ye

When 5G telecommunication becomes a standardized and widely used communication medium, it must be implemented in coherence with certain 5G network standards and requirements. One such requirement is a Subscription Concealed Identifier called SUCI. SUCI prevents the exposure of international mobile subscriber identity (IMSI), which was a vulnerability in previous generation mobile telecommunication networks. Unlike IMSI, SUCI is encrypted and transmitted using a symmetric key cryptographic algorithm, to prevent the aforementioned vulnerabilities. However, for the first terminal to be encrypted, it is necessary to exchange a key with the home network, and this key exchange for SUCI encryption is performed through the Elliptic Curve Integrated Encryption Scheme (ECIES) key exchange algorithm, which is a public-key encryption scheme. However, ECIES uses more computing resources compared to a symmetric key cryptographic algorithm. Additionally, for 5G Subscriber Identity Deconcealing Function (SIDF) to satisfy the massive machine-type communication (mMTC) requirements of 5G, it is necessary to decrypt at least a million SUCIs within a short time. This puts a great burden on the 5G home network to provide the mMTC service for IoT. Therefore, in this paper, we propose a method of constructing 5G SIDF in an mMTC IoT environment. A key method of the proposed 5G SIDF configuration is the use of GPUs. This proposal was aimed at reducing the load in the mMTC environment by performing parallel processing of all cryptographic operations performed in the SIDF using a GPU. In particular, we focused on parallelization of public-key encryption algorithms. In addition, we also compared the method proposed in this paper through a survey of various 5G security products.

1. Introduction

The fifth-generation (5G) telecommunication technology needs to follow the IMT-2020 Standard (International Mobile Telecommunications-2020) [1] requirements defined by the ITU-R (ITU-Radio communication sector) of the International Telecommunication Sector in 2015 for 5G networks. This standard defines the evolution of telecommunication with respect to various technical requirements. Based on this, the 3rd Generation Partnership Project (3GPP), an international standardization organization for mobile communications, standardized 5G NR (New Radio) in 2016 and has been contributing to the commercialization of 5G technology until Release 17 as of 2021. 5G telecommunication has caused various changes in the network and security fields. The three requirements called enhanced mobile broadband (eMBB), massive machine-type communication (mMTC), and ultrareliable low-latency communication (URLLC) govern the 5G telecommunication as it revolutionizes existing mobile communication. In other words, 5G telecommunication networks will provide services while satisfying one of these three requirements. As such, with the advent of the three 5G services, the communication environment of the Internet of Things

(IoT) has been further developed. Among them, this paper focused on the mMTC service. The mMTC service area is designed for massive IoT deployments using large numbers of low-power devices to regularly transmit small amounts of data. With the increasing popularity of intelligent transportation, smart city, etc., it is envisioned that the number of IoT devices will reach 75 billion by 2025, which is much larger than the number of the mobile phone users. To provide wireless connectivity to such a large number of devices by the time IoT comes to fruition, 3GPP has identified mMTC as one of the three main use cases of the 5G wireless systems [1]. As such, providing 5G mMTC service is very positive for the IoT environment.

However, 5G mMTC also has a big problem. From 5G mobile communication, the public-key encryption method is applied in the Authentication and Key Agreement (AKA) process of the 5G terminal (UE). When a public-key encryption scheme is applied, both the UE and the home network become a great burden. Particularly from the perspective of the network, it should be able to handle the simultaneous access and authentication process of up to 1,000,000 UEs to support the mMTC service. In this paper, we focused on the 5G Subscriber Identity Deconcealing Function (SIDF), which is in charge of public-key cryptography. This function is always used for initial access authentication to the terminal, and a load may occur if many terminals suddenly try to access it, such as in the mMTC environment. Unlike the core network up to 4G, the 5G's core network is designed through a software-based core structure; therefore, each network function (NF) can be configured as a module and the core network can be configured to meet various requirements. Therefore, in this paper, we propose a parallelization scheme using graphics processing unit (GPU) to improve the ECIES operation speed of the 5G telecommunication core network that satisfies mMTC.

The contributions to this paper are as follows:

2. Proposal of X25519 ECIES Decryption Parallel Processing Using GPU

This paper proposes to speed up the ECIES operation used in 5G AKA from the perspective of the home network. Various existing ECC speeding studies mentioned in this paper suggest a method for speeding up fixed scalar computation. That method is ultimately only available for the ECIES encryption method. However, in this study, to speed up ECIES decryption, we propose a method of predividing a fixed scalar and speeding up the random scalar computation using GPU.

3. Suggestion of 5G SIDF Function Construction Using GPU for mMTC IoT Environment

This paper presents a method to configure the SIDF function using GPU to speed up 5G AKA in the mMTC environment. The functions of the 5G core network are implemented the in software to freely configure the network. Therefore, in this paper, when the 5G home network SIDF performs the ECIES decryption operation, a new approach is introduced to quickly perform the 5G key agreement process.

4. Comparison with Commercial 5G Security Product

To show the superiority of the proposed SIDF function, we survey and compare cryptography and security function products used by various 5G carriers.

5. Background

5.1. 5G SUCI. Until the implementation of 4G, international mobile subscriber identity (IMSI) was used as a subscriber identifier. The corresponding identifier was transmitted to the network upon initial access to the terminal. At this time, a vulnerability was exposed in the wireless section [2, 3]. This vulnerability became a problem in the development of subsequent telecommunication standards. To solve it, various identifiers have been encrypted and transmitted starting from the implementation of 5G telecommunication. Consequently, SUCI has been developed [4–6]. SUCI is a concept first introduced in 5G telecommunication networks and is a value transmitted by concealing (encrypting) the identifier in the UE to prevent identifier's exposure when transmitted to the wireless section. Therefore, even if the corresponding value is captured in the wireless section, the terminal identifier value is not exposed because it is encrypted. Figure 1 demonstrates the structure of SUCI. Among the SUCI fields, the "Protection Scheme ID" field specifies how to conceal the identifiers with SUCI, in which 0×0 denotes "NULL = scheme", 0×1 denotes "ECIES Profile A (curve 25519)", and 0×2 denotes "ECIES Profile B (secp256r1)." Subsequently, to conceal an identifier, a public-key encryption scheme using an elliptic curve cryptography is used.

5.2. ECIES in 5G Security and X25519. ECIES is a hybrid encryption scheme that encrypts data using symmetric keys created through an elliptic curve cryptography-based key exchange method, a public-key cryptography method [7]. In 5G, ECIES is used to encrypt the MSIN in the IMSI and securely transmit it to the 5G home network (5G HN). Therefore, the UE side of 5G communication uses the ECIES encryption method as shown in Figure 2, and the SIDF of 5G HN uses the ECIES decryption method described in Figure 3. First, the UE must generate a temporary encryption key to encrypt the MSIN. To this end, a key exchange method using elliptic curves is used. The UE generates K different keys randomly for each access session. Then, the key material of the ephemeral shared key, namely, ephemeral public key R, is calculated. The shared value Z is generated using K and the home network public key Q. Then, encryption and MAC keys are constructed using the generated R and Z as inputs for the key-inducing function. Subsequently, the MSIN in IMSI is encrypted using the AES-128-CTR algorithm, and a MAC value (HMAC-SHA-256) is generated to verify that the correct value is encrypted. When the home network's UDM receives a SUCI signal from AUSF

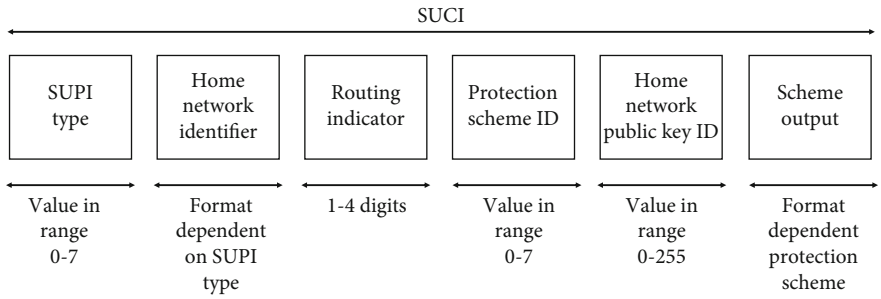

Figure 1: SUCI structure: SUCI is a concept first introduced in 5G telecommunication networks and is a value transmitted by concealing (encrypting) the identifier in the UE to prevent identifier's exposure when transmitted to the wireless section.

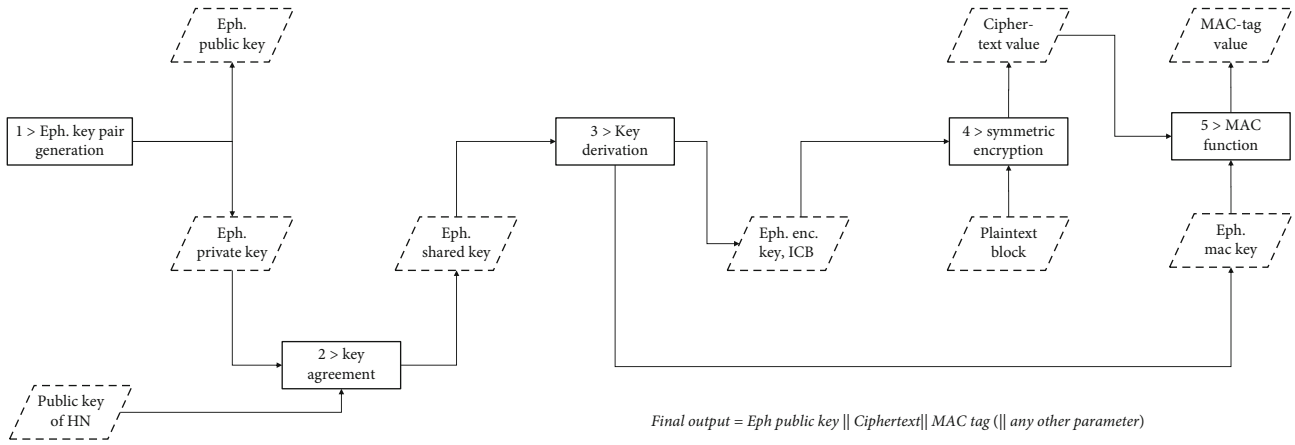

Figure 2: ECIES encryption process on UE side: in the UE (USIM), the ECIES encryption process is performed. In this step, a 128-bit encryption key and a 128-bit mac key used to generate SUCI are generated. Thereafter, the MSIN, which is used in 4G subscriber identifier, is encrypted to generate SUCI.

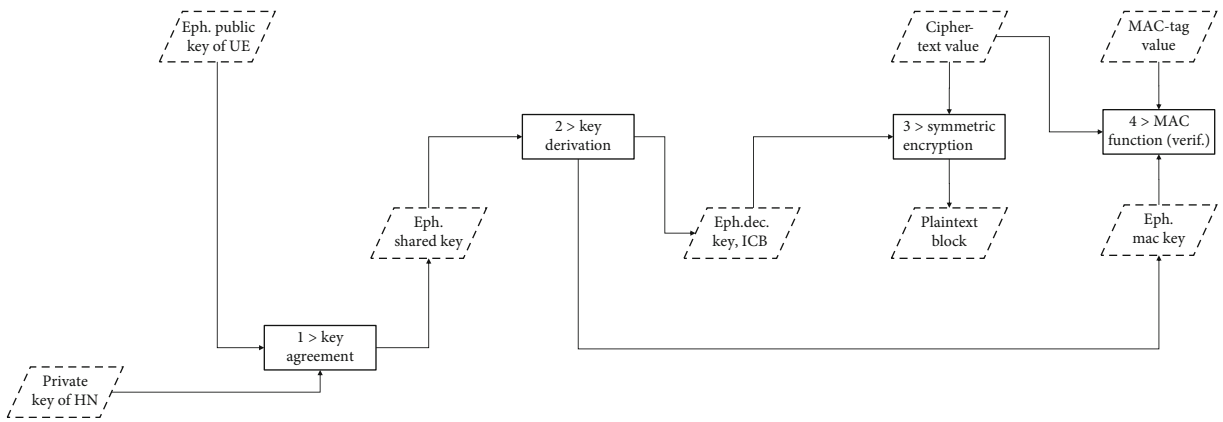

Figure 3: ECIES decryption process on home network side: in the SIDF in the home network, the ECIES decryption step is performed. In this step, a 128-bit encryption key and a 128-bit mac key are generated to decrypt SUCI. Thereafter, the MSIN is generated by decoding the SUCI transmitted from the UE.

on the serving network, the SIDF function is called. SIDF uses the received ECC ephemeral public key of the UE and private key of the home network to generate an ephemeral shared key. Next, the SIDF decrypts the MSIN that was encrypted and transmitted. After that, the KDF function generates the encryption and MAC keys and deconceals the SUCI of the UE. After this process, the HN determines the AKA method to be utilized with the UE based on deconcealed SUCI before its execution.

Elliptic curve cryptography (ECC), a public-key cryptography scheme used in the key agreement process, was independently introduced by Koblitz and Miller in 1985 [8, 9], addressing the existing discrete logarithm problem (DLP) with a much shorter key length and faster speed.

Therefore, it is used for key agreement or digital signature tasks in various small IoT environments or smart card environments. Because of its advantages, ECC was used for the first time in a communication network with the adoption of 5G technology. Because the ECC curve equation varies according to the definition of curves and parameters, it is important to determine which curve is used among the objects accurately. Therefore, in 5G telecommunication networks, profiles A and B are defined a priori. Profile A is the X25519 parameter on Curve25519 [10], and profile B is the secp256r1 curve [11].

5.3. Security Availability for 5G Services. Unlike 4G telecommunication security standards, the 5G security standards are diversified through authentication and key agreement methods, key hierarchy is more refined, and a public-key cryptographic scheme called Elliptic Curve Integrated Encryption Scheme (ECIES) is applied for the first time. ECIES has not been considered in the 4G security standard because of a required longer key length and slow operation compared to the symmetric key cryptography. Consequently, 5G telecommunication must satisfy both network and security requirements, and this is a challenging task.

In the eMBB service, improved transmission speed, maximum transmission speed, and terminal mobility are provided by improving the service quality of the existing 4G mobile broadband (MBB), and the URLLC service refers to a communication service that has a very high reliable user data transmission with a very short delay time. Therefore, in order for the eMBB service to transmit high-capacity data and provide maximum transmission speed, it is essential to speed up the data plane, signal data encryption/decryption, and integrity functions among 5G security technologies. URLLC service is particularly needed to speed up encryption/decryption functions, mainly the integrity functions in the data plane's wireless section. However, even in 3GPP TS 33.501, the 5G security standard issued by 3GPP, encryption/decryption and integrity functions of user data and signalling data are mostly classified as "optional" [4]. In addition, the network overhead caused by using the encryption/decryption and the integrity functions can be a great burden on the mobile operator. Therefore, many mobile carriers adopt the NULL-scheme, where they do not provide encryption/decryption and integrity functions. However, as mentioned above, security vulnerabilities are found because 5G mobile communication is used in various environments, so it is essential to provide a cryptographic algorithm availability.

In this paper, we want to focus on the mMTC environment in addition to the above two scenarios. Unlike eMBB, mMTC targets IoT devices that exchange relatively low-capacity data at low speed. The mMTC service defines the maximum number of terminals accessing the network as 1 million per unit area (1 square kilometer). For the 1 million devices to connect to the network and communicate with each other, it is absolutely necessary to speed up the initial authentication procedure on the core network of 5G encryption technology. However, the biggest problem is that, unlike eMBB and URLLC, the public-key encryption method using ECIES must perform up to 1,000,000 times in the initial authentication process. Therefore, to provide initial authentication and key agreement functions in the mMTC environment, the fastest and most accurate ECIES public-key operation is required for terminals and 5G core networks.

Secondly, the mobile communication technology defined in international standards is used for more than one generation; therefore, one has to take into consideration the restrictions associated with each generation. Finally, the upcoming 6G telecommunication technology is expected to be governed by faster dynamics than those of the current 5G telecommunication requirements, with the aim of expanding to more connected devices. In this case, faster and more accurate cryptographic operations are required. In 5G telecommunication environments, because the number of small terminals, using the universal subscriber identity module (USIM), in the existing IoT environment increases rapidly, various entities in the 5G core network must be able to simultaneously process transmissions from numerous terminals. In particular, the mMTC requirements must cover up to 1,000,000 devices per square kilometer.

5.4. GPGPU. Many personal computers use GPUs for computer graphics operations and screen resolution processing. A method for achieving parallel computing using the GPU hardware for various purposes is called General-Purpose Computing on Graphic Process Unit (GPGPU). In particular, NVIDIA has developed so-called "Compute Unified Device Architecture" (CUDA), a general-purpose parallel programming framework that uses a programming language, such as C language, for compilation and execution of GPU programming in 2007. CUDA framework can be used in both Windows and Linux environments. Hence, many application developers can implement various functions, such as AI, accelerated computing, and cryptocurrency, using NIVIDA GPUs [12]. The structure of NVIDIA GPUs using the CUDA framework has been gradually developed. The structure of a physical GPU consists of multiple streaming multiprocessors (SMs) and streaming processors (SPs) that execute the program codes in each SM. SPs have the same structure as a CUDA core in terms of hardware. The logical GPU structure is organized in the order of threads-blocks-grids, as shown in Figure 4 [13]. A thread is the minimum command processing unit, and usually, 32 threads are gathered to form one warp, which is the smallest unit of execution. The threads gathered in a warp form a block, and the blocks form a grid. Each logical structure depends on the resources of the GPU, while the warp size is usually composed of a multiple of 32 threads. Therefore, the warp unit is the most important when performing regular parallel programming.

To leverage the GPU effectively, one needs to understand how a GPU works. The GPU bundles several threads and executes the same commands in one warp unit. Therefore, it should be implemented to ensure control divergence does not occur, which is achieved by minimizing branch statements. Moreover, because the GPU resource that has the greatest influence on block configuration per SM is the registers, to increase occupancy, the

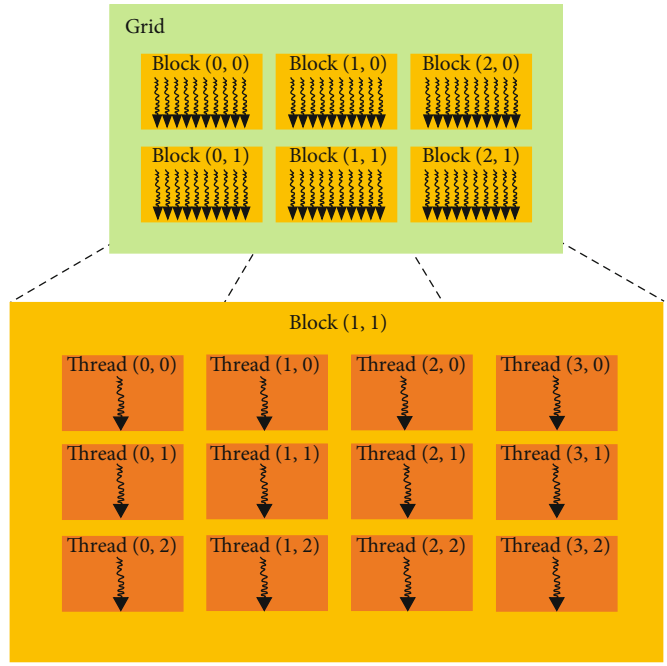

Figure 4: Thread hierarchy in CUDA programming: this figure is the logical structure of CUDA programming. As shown in the figure, it has a structure in which the kernel function operates simultaneously through the grid-block-thread stage. In this paper, the ECIES decryption kernel function operates through the structure as well.

number of active blocks should be increased by optimizing the registers used per thread.

6. Related Work

Cryptographic algorithms are used in various environments, such as small IoT environments, smart cards, and parallel computing environments. When providing security services by implementing cryptographic algorithms in various environments, because the physical capabilities of each environment are different, various methods are used to increase the operation speed or memory efficiency in environments with insufficient memory. Particularly, there are numerous restrictions when providing a security service using public-key cryptography algorithms. Many studies were aimed at improving speed and memory efficiency because basic operations, such as big-number operations of public-key cryptography algorithms consume more memory and are slower than symmetric key cryptography algorithms. In addition, public-key cryptosystems based on discrete logarithmic problems, such as RSA, DSA, and DH, are disadvantageous in terms of speed and memory efficiency compared to symmetric key cryptosystems. Therefore, ECC [7, 8], which provides security services such as digital signature and key exchange services, can outperform the existing public-key encryption system due to its short key length and has become popular in various environments, including small IoT applications. Therefore, most researches are focused on high-speed implementations in small chips with insufficient computing resources [14–17].

6.1. ECC Parallel Implementation Study Using GPU. As the GPGPU era started, cryptographers worldwide have conducted research to improve the speed of scalar multiplication operations in ECC algorithms using GPUs, defining a new computing environment. [18, 19] used the CUDA framework and obtained parallelization results for fixed scalar multiplication and fixed point using a single thread. However, in [18], the parallelization results based on discrete logarithm problems were mainly implemented. Thus, the ECC algorithm utilized only the NIST P-224 curve. On the other hand, [19] focused on the high-speed implementation of GMP-ECM using the Edward elliptic curve in various GPUs. Furthermore, in [20], high-speed implementation of ECC scalar multiplication through GPU programming was studied. This study demonstrated the parallelization results of various elliptic curves over gf (2^m). Unlike the previous studies, the performance was optimized by calculating the random and fixed scalar products differently. In [21], by optimizing the algorithm at the assembly level using CUDA Parallel Thread Execution (PTX) instruction set architecture (ISA) using GPUs in X25519 key agreement, an efficient method for modular addition and multiplication of Curve25519/448 and concise reduction arithmetic is proposed. In this paper, unlike [20], the ECC modular multiplication operation is configured as a PTX operation, and scalar multiplication of the base point and scalar multiplication of an unknown point multiplication are all implemented by using one thread for each scalar multiplication. Finally, in [22], the DPF-ECC framework was presented to accelerate the ECC system by maximizing the double precision floating point (DPF) computing power. Unlike the above papers, this

paper proposes a new framework that applies DPF operation to ECC. In particular, it showed up to 3 times the amount of throughput compared to the previous research results on the Edwards25519/448 curve as well as in a specific prime number computation using GPU.

6.2. 5G Core Network Security Solution Survey. Implementing security functions within 5G systems is standardized to a limited extent by 3GPP. Therefore, 5G equipment manufacturers need to implement 5G network functions such as AUSF, ARPF, SIDF, and UDM related to subscription authentication when connecting to a 5G UE network and implement various encryption keys and algorithms generated as a result of UE authentication. However, 5 g network solutions will be subjected to various stress tests due to the growth of 5G and industrial digitalization. Accordingly, various equipment manufacturers must provide an additional level of security to software-only solutions for security functions (subscription authentication, encryption, and key matching) standardized by 3GPP when building 5G network equipment and increasing the demand for network capacity. In particular, equipment manufacturers provide integrated management by building security solutions based on the cloud to satisfy the 5G characteristics of SDN, network slicing, and 5G NF design ideas. In this section, we would like to mention the solutions of two representative companies, Ericsson and Nokia. Ericsson, a leading 5G equipment company, configured Cloud Core Subscription Manager (CCSM) by combining UDM, HSS, AUSF, and EIR as a 5G authentication solution to respond to this problem. This CCSM is composed of a cloud-native 3GPP network function. The solution provides an authentication security module to provide 5G security functions standardized by 3GPP and provides various security and encryption functions by linking 3GPP ARPF and HSM provided by Thales [23]. In addition, Nokia, a leading 5G equipment company, has been engaged in 3GPP 5G standardization work and has been engaged in various activities such as 5G security function proposals and patent registration. In particular, in order to respond to the problems of 5G security threats, UDM, HSS, AUSF, HLR, EIR, etc. are bundled together as a 5G authentication solution, and Subscriber Data Management (SDM) and the integrated authentication solution Authentication, Authorization and Accounting (AAA) are integrated into 5G configured in the network [24].

6.3. 5G SIDF Open-Source Project Review. In Open5gs [25], an open-source 5G network project, SIDF is not implemented separately but is included in the UDM function. Moreover, there was no separate implementation of public-key cryptography, except for the implementation (embedded) of the null type. Looking at the SIDF implementation in Free5GC [26], another 5G network open-source project, the SIDF function was not implemented separately, similar to Open5gs. However, it differed from Open5gs regarding the ECC algorithm implementation for SUCI profiles A and B, which was implemented in UDM to deconceal SUCI. Because Free5GC is a 5G network-based open-source project, it has a format that is called every time a single HTTP message is transmitted. Particularly, the encryption algorithm was implemented in Go Language with no specific high-speed technique applied [27]. Looking at OpenAirInterface5g [28], another 5G network open-source project, the OpenAirInterface5g project is divided into 5G RAN and 5G CN. In particular, the OpenAirInterface5g CN project was aimed at making a CN stack fully compatible with 3gpp. The OpenAirInterface5g CN project is currently in progress up to version 1.3 and includes various 5G NFs such as AMF, AUSF, and NRF. The most prominent feature with other open sources is that C and C++ are used as the major languages, so there is an advantage for 5G network configurators to use these open sources in the actual 5G environment. However, as in the above two open-source projects, the SIDF function is not implemented separately; in particular, the method of handling SUCI within AUSF and UDM is not specified, and it is hard coded in the 5G RAN project. Like [25, 26, 28], the 5G core network has the characteristic of freely configuring the network by implementing each NF in the software. This is not only a 5G design idea but has several advantages. However, when composing SIDF with software alone, it was judged that the ECIES operation speed was slow like the two open sources, or it could not satisfy the mMTC requirements. Subsequently, in our study, when the 5G home network SIDF performs the ECIES decryption operation, instead of using the methods introduced in the abovementioned studies, we introduce a method that quickly performs the 5G key agreement process with a novel approach.

7. Suggested Method: Parallel ECIES Decryption Using GPU

7.1. Parallel 5G SIDF Construction Using a GPU. In the abovementioned studies, high-speed implementations of ECC scalar multiplication were studied in various ways. However, in the SIDF framework for 5G telecommunication networks, the ECIES decryption must be performed differently than the ECIES encryption step that performs the fixed point and random scalar multiplication operations similar to those performed in the abovementioned studies. The SIDF of the home network performing the ECIES decryption step must perform a fixed scalar product (home network private key, d) and assign it to a random point (ephemeral public key, R) for network access of one terminal. In addition, when many terminals are accessed simultaneously, it is necessary to multiply a fixed scalar by a number of random points. Therefore, in the ECIES decryption step, the random scalar multiplication operation on various fixed points cannot be performed similarly to those proposed above. Here, the 5G telecommunication home network uses a GPU to speed up the random point and fixed scalar multiplication operations, which requires the most computation time and resources during the ECIES decryption process. This method is expected to satisfy the mMTC requirement when numerous terminals access the 5G network. The basic configuration of the proposed 5G SIDF is depicted in Figure 5.

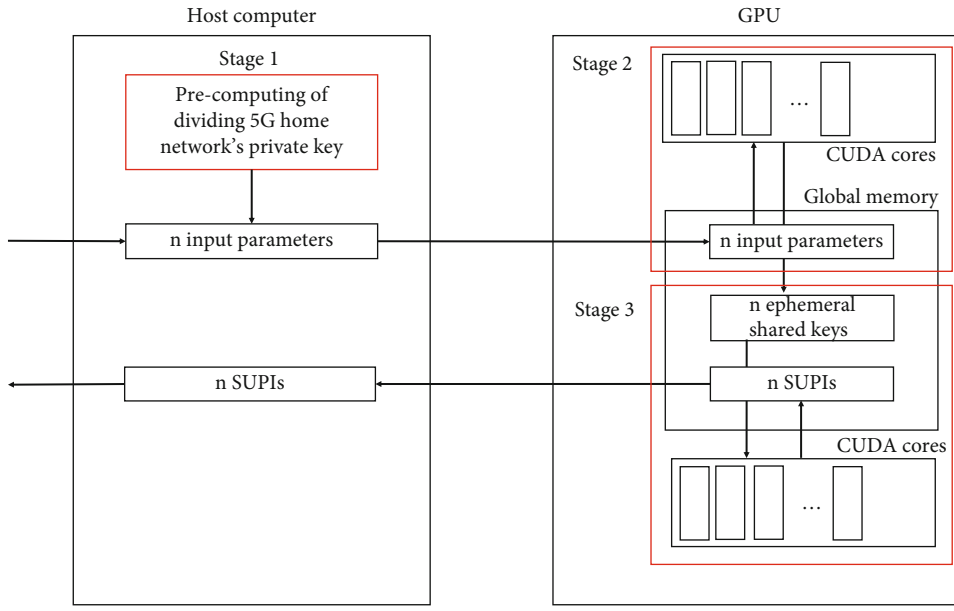

Figure 5: Suggested model of 5G SIDF using GPU: it is suggested to configure 5G SIDF in mMTC environment as shown in the diagram. The SIDF proposed in this paper consists of a host computer and GPU. In particular, in the GPU, the ECIES decryption step and the SUCI decryption step are performed using parallel computing technology.

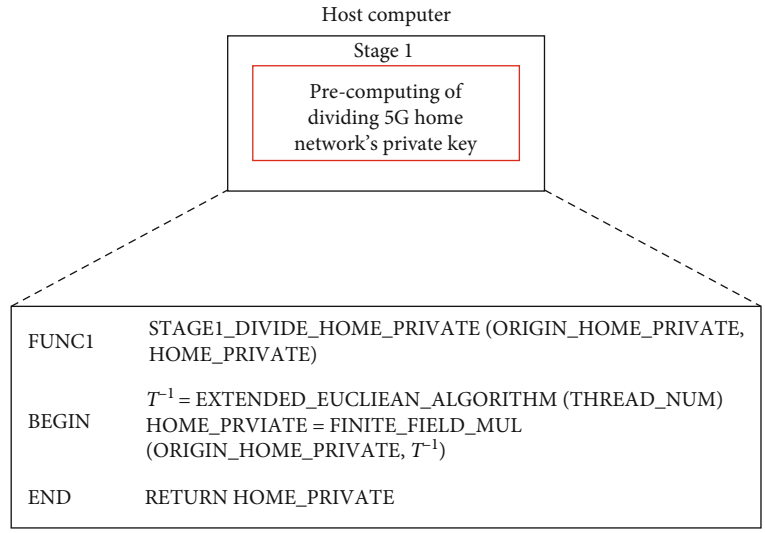

Figure 6: Stage 1 procedure: stage 1 performs precalculation. This step works on the host PC. In the precomputation stage, the private key of HN is divided through a finite field operation.

The SIDF parallelization structure using the GPU is divided into three stages.

7.1.1. Stage 1. Stage 1 is the precomputation part that divides the private key of HN according to the number of threads on the GPU. This method was not found in the previous studies. In the ECIES decryption stage, public key and private key pair of HN are a fixed value that continues to be used as long as the validity period has expired and is not changed according to Telco's policy. Therefore, the private key value of the HN is fixed and used whenever numerous UEs access the network. In addition, in SIDF, an operation of multiplying the private key of HN at different random points transmitted by numerous UEs is performed. Therefore, in this paper, the scalar, which is a fixed value to be used for parallel operation of the GPU, is divided and used through stage 1, the precomputation stage.

Stage 1 operates like FUNC1 in Figure 6. FUNC1 operates on the host PC. FUNC1 basically receives private key of HN. The private key of HN is a scalar value within the elliptic curve finite field. Therefore, in order to speed up the scalar multiplication operation performed during ECIES decryption, the private key of HN is divided by the GPU thread value (T). The calculation should be done on the finite filed using elliptic curve. However, in the finite field, the division operation is not defined separately, so the

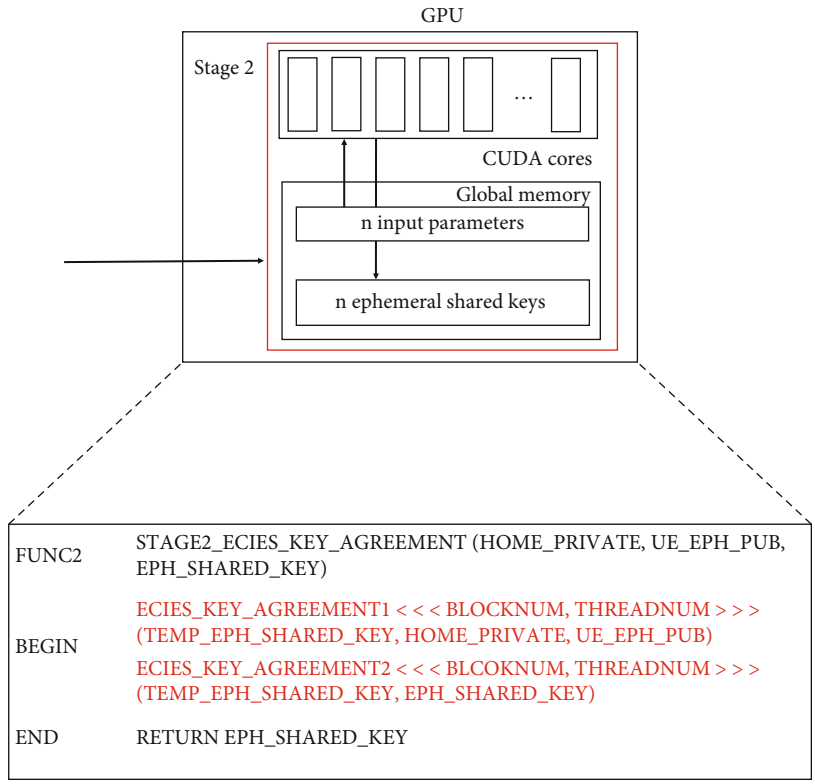

Figure 7: Stage 2 procedure: stage 2 calculates the ephemeral shared key. These steps run on the GPU. Stage 2 receives the value generated in stage 1 and the random point (ephemeral public key) received from the UE and performs ECC scalar multiplication operation.

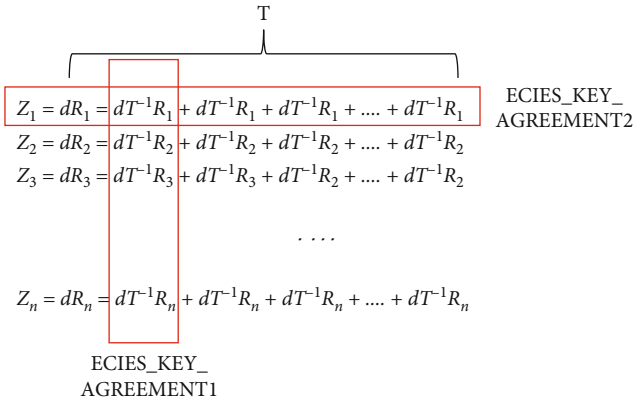

Figure 8: Stage 2 kernel function concept formula: in fact, the contents that operate in the kernel function of stage 2 are expressed as a formula.

method of multiplying the value to be divided by the inverse must be used. Therefore, it is necessary to find the inverse (T^{-1}) of the value T in the finite field. However, the GPU thread value is fixed depending on the GPU used. And in general, GPU thread size is a multiple of 32. Therefore, the T^{-1} is also a fixed value. Therefore, the precalculation stage, stage 1, is used only when the key pair of the HN is changed or the corresponding value is deleted because the power is turned off and generally does not operate when the UEs access.

The reason for performing stage 1, the precomputation stage, is that the scalar multiplication operation speed becomes slower as the length of the ECC private key increases, as demonstrated in various studies [29, 30]. The reason scalar multiplication takes longer as the ECC key length increases is because the key length is related to the size of the point. As a result, stage 1 makes it possible to reduce the amount of elliptic curve scalar product computation in stage 2 running on GPU.

7.1.2. Stage 2. Stage 2 calculates the ephemeral shared key Z. This step is the most essential part of the thesis. The GPU operates in a way that many cores simultaneously process the same operation with one source code. For example, if a GPU is configured to perform an addition operation, multiple cores derive a result by only performing an addition operation on a given input parameter. Therefore, an efficient configuration of GPU resources is required. This step consists of an ECC scalar multiplication operation part and ECC point addition operation part.

First, in the ECC scalar multiplication operation part, the UE's ephemeral public key transmitted from the UE is multiplied by the private key of HN divided in advance in stage 1. At this time, because private key of HN may not be a multiple of the operation unit, it is not easy to ensure a perfect resource. However, since the number of UEs connected to

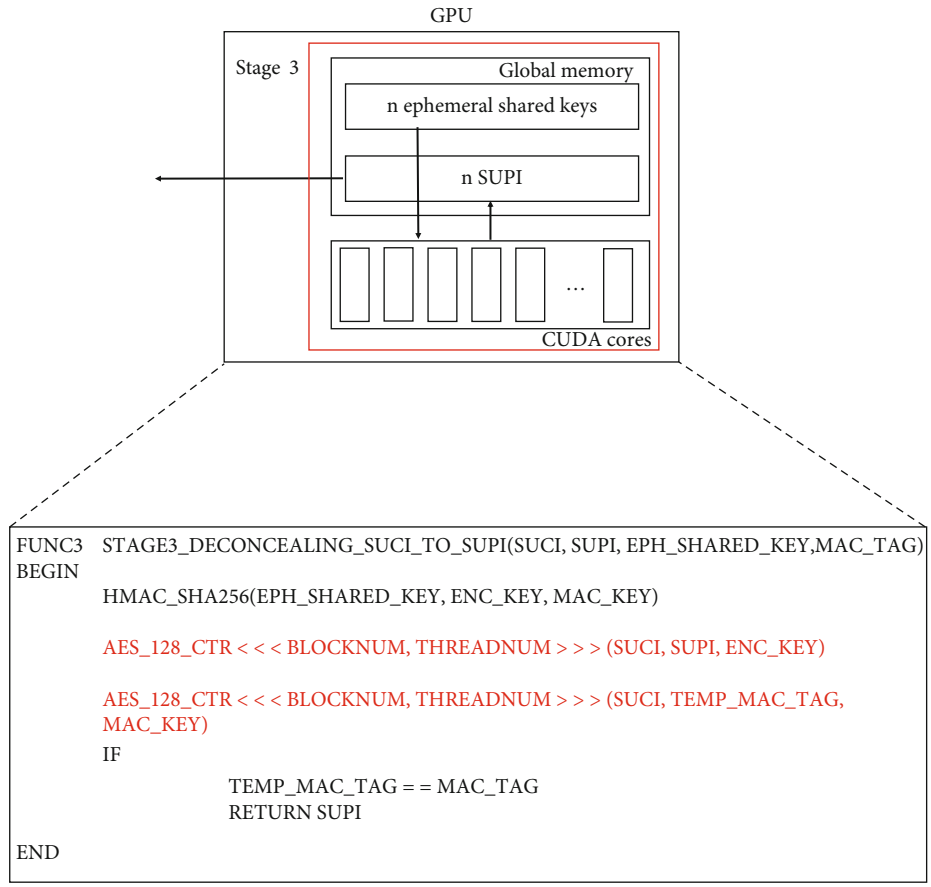

Figure 9: Stage 3 procedure: obtain encryption key and MAC key using the ephemeral shared key, the result of step 2 in step 3. The block cipher (AES) operation runs on the GPU.

the HN is exceedingly large, performing scalar multiplication by dividing in advance in stage 1 achieves better efficiency. In addition, because the HN has one public key pair that is used for a long time, the SIDF can be fixed and configured without changes for each UE connection. In this way, the scalar multiplication operation part multiplies R_i transmitted by each UE by dT^{-1}. The corresponding value is the same as the value obtain by dividing the ephemeral shared key by T. This part works on ECIES_KEY_AGREEMENT1 in Figure 7.

ECC points multiplied in this manner must be added together once again to match the UE index to form an accurate ephemeral shared key. Therefore, the ECC points calculated for each UE are stored after the scalar multiplication; then, the ECC point addition operation is performed. Because the ECC point addition operation is invariant and the number of points added for each UE is fixed, it can proceed without processing each UE approach separately. If the precomputation step is omitted, the Z value can be obtained directly using the scalar multiplication operation by inputting only the existing d without the divided private key. The corresponding contents works in ECIES_KEY_AGGREMENT2 in Figure 7. And all this process satisfies the equation in Figure 8. FUNC2 operates on the GPU as shown in Figures 5 and 7.

7.1.3. Stage 3. In stage 3, the ephemeral shared key Z value, which is the result of stage 2, is processed using the key derivation function HMAC-SHA256 to obtain the encryption and MAC keys. Then, the correct SUCI value is obtained using the AES-128-CTR decryption algorithm and verifying the MAC function. Unlike the other stages, the third stage is not a public-key cryptography operation but a symmetric key block cryptography operation. This stage has a simpler operation process than the above two processes; hence, its speed is rapid. Consequently, the operation can be performed on the CPU or GPU. If the third stage is configured using a GPU, in this case, the values must be derived for each n number of UEs, and GPU source code must be divided considering the entire key derivation and symmetric key block cipher. The last step can also be confirmed with FUNC3 in Figure 9.

When configuring the proposed 5G SIDF, the biggest difficulty in the GPU configuration is the GPU's operation method, as mentioned in Background. Kernel functions operate in warp units in all SMs within the GPU in the GPU. This is the biggest advantage and characteristic when implementing a cryptographic algorithm using GPU, but it is also the biggest disadvantage. It is important that as many warps as possible are simultaneously performing work on all SMs that are components of the GPU. When an ECIES

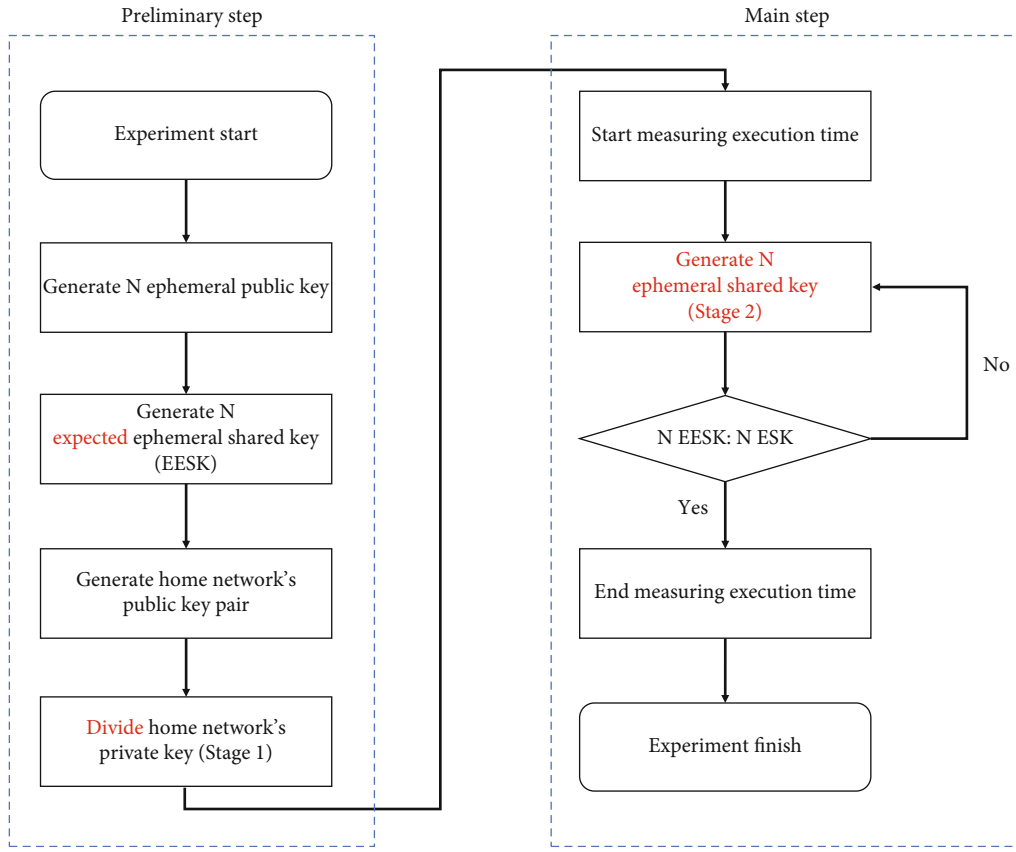

Figure 10: Experimental flowchart: the experiment was configured like the corresponding flowchart. The experiment is largely divided into a preliminary step and a main step. In the preliminary step, all the preparation values necessary for the experiment are generated. Therefore, EESK is generated, and the process of dividing the private key of the home network is performed. In the main step, N ESKs are generated, compared with the EESKs, and the total time is measured.

decryption request does not come with a warp multiple size, the kernel function is terminated first in a specific SM, and the final end time becomes the end time when the warp multiple size is the size. Therefore, it is necessary to consider how to configure the SIDF when a request is made rather than taking a multiple of the number of warps. The configuration method presented in this paper performs the ECIES decryption process without filling the number of warps even when a request is received, which is not a multiple of the number of warps. The second difficulty is selecting GPU equipment and implementing GPU kernel functions. GPUs are generally graphics devices and cannot be used on their own. Therefore, in the 5G network, when AUSF or SIDF receives an ECIES decryption command, there must be a host PC that can give a command to the GPU again. Also, as mentioned above, NVIDIA GPU was used in this paper. However, the CUDA library cannot be used when using an AMD GPU. Therefore, it has the disadvantage of not using all GPUs. However, the CUDA library was only available in C and C++ as the initial development languages, and of course, the C language is optimized for implementation. However, recently, by accommodating various languages such as go, python, and Fortran, network design and interoperability with various open-source projects are good. Therefore, if you use NVIDIA's GPU, there is no difficulty using the GPU in the 5G network.

Table 1: Device A target platform configuration.

Device A	Contents
CPU	Intel Core i7-9700
Clock	3.00G Hz (800~4700 MHz)
Core/thread	8 core/8 thread
RAM	DDR4 16 GB X 2 (32 GB)
OS	Ubuntu 18.04.3 LTS
Compiler	gcc 7.4.0

Table 2: Device B target platform configuration.

Device B	Contents
GPU	NVIDIA GeForce GTX 1060
Architecture	Pascal
CUDA driver version	11.3
CUDA capability	6.1
CUDA cores	1280
Warp size	32

8. Experiment

8.1. Experimental Setup. By implementing the SIDF structure proposed in Chapter 3, we present the performance values of the proposed method. The experiment in this study is performed only for the X25519 key agreement process using Curve25519, which denotes stage 2, 5G ECIES profile A, except for the key derivation function and decryption process of the symmetric key block cipher algorithm. The reason for experimenting only with stage 2 is that it is based on public-key cryptography operation, while stage 3 is based on symmetric key block cipher operation. Symmetric key block cipher has a simpler operation compared to public-key cryptography. Thus, the operation speed is exceedingly fast compared to public-key cryptography. However, many studies have been published on this subject. Therefore, separate experiments were not conducted in this paper [31–33]. In addition, the reason for testing only profile A and not B is that the key agreement process using profile A has a faster basic operation speed and uses less memory compared to the key agreement process using the Secp256r1 curve, which denotes profile B [34–36]. Therefore, profile A is likely used when many mobile carriers implement ECIES as an actual key exchange algorithm without using the NULL-scheme. The cryptography algorithm used in the experiment is a modified version of OpenSSL V1.1.1g [37]. There are various implementation methods for the basic operation of Curve25519, which may vary in speed, source code size, and memory usage depending on the implementation method [34, 38–40]. Therefore, to achieve an unbiased experiment, the Curve 25519 source code in OpenSSL, which is used worldwide, was used as the basic source code.

The experimental process is carried out according to the procedure described in Figure 10. In the preliminary experimental stage, basic information regarding all experiments is generated. Particularly, all values used as input parameters are generated in the ECIES decryption process, which is the main stage. When the UE accesses the 5G home network, the UE ephemeral public key is generated along with SUCI. Simultaneously, (d, Q), a public key pair used by the home network to perform public-key operations is generated. To generate the expected ephemeral shared key, which is the result of the main step, the ECIES encryption step is performed for each terminal. In addition, the precomputation, which divides the home private key according to the number of threads, is also carried out in the corresponding stage. After the preliminary phase, the main experimental phase proceeds.

During the main experiment, as the number of UEs attempting to access the home network increases, the home network load is examined and the effectiveness of the proposed SIDF configuration that uses ECIES decoding calculation for each experimental device is tested. As mentioned above, the number of UE terminals in this experiment, that is, the number of ECIES decryption calculations, is set to 2^{10}, 2^{15}, 2^{17}, 2^{19}, and 2^{20} for each iteration. Each device receives d and Q generated in stage 1, the number of UEs attempting to access the home network, and the ephemeral public key of each UE. Each device calculates the ephemeral shared key through the ECIES decryption using only the received value and compares it to the expected ephemeral shared key generated in stage 1. All calculation times are measured if all values match when UE is compared. In the process described above, Device A sequentially calculates up to ephemeral shared keys, and device B calculates up to ephemeral shared keys through GPU parallelization based on the calculations performed in the precomputation process. The experiment conducted using device A is called test A, and the experiments conducted using devices B and C are called tests B and C, respectively.

Table 3: Device C target platform configuration.

Device C	Contents
GPU	TITAN Xp
Architecture	Pascal
CUDA driver version	10.2
CUDA capability	6.1
CUDA cores	3840
Warp size	32

Experiments were performed using the three devices specified in Tables 1–3. The first device (device A) is configured using a general CPU. The second and third equipment (devices B and C) are GPU configured for testing the proposed implementation method and SIDF. The resources of devices B and C only differ in the global memory size and CUDA core, while the rest of the GPU resources remain mostly the same. The experimental method is stated as follows: first, perform the experimental preliminary steps on the host PC (device A).

8.2. Experiment Results. The superiority of the proposed SIDF configuration is demonstrated by comparing the results of the abovementioned experiments. In the test using the three devices, the generation times for 2^{10}, 2^{15}, 2^{17}, 2^{19} and 2^{20} ephemeral shared keys were measured, respectively. Moreover, because the experimental results measure the time needed to generate ephemeral shared keys, each generation time is different depending on the length or value of the input parameter. Therefore, to increase the measurement accuracy, each test was performed twenty times, where each generation time was measured in microseconds (ms). The values reported in Tables 4–6 are the time it took to generate each temporary shared key in each environment. Particularly, in the experiment where each ephemeral shared key is generated through a GPU parallelization, the time taken to move the memory from the host device to the GPU device was measured. In test A, generating 2^{10} ephemeral shared keys took approximately 79.00 ms, 2^{17} ephemeral shared keys took approximately 10108.57 ms, and 2^{20} ephemeral shared keys took approximately 80879.87 ms. In other words, it took approximately 81 sec for the SIDF to perform ECIES decoding when roughly 1,000,000 UEs attempted to access it simultaneously. However, when a GPU is used, the SIDF processing speed increases dramatically. In test B, it took approximately 8.43, 227.96, and 1788.13 ms to generate 2^{10}, $2^{17}2^{17}$ and 2^{20} ephemeral shared keys, respectively.

Table 4: Results of test A (ms).

Iterate count	Number of ephemeral shared keys				
	2^{10}	2^{15}	2^{17}	2^{19}	2^{20}
1	78.97	2527.04	10109.66	40437.55	80880.75
2	79.03	2527.30	10110.96	40438.98	80877.55
3	78.98	2526.82	10106.96	40438.34	80877.09
4	78.98	2527.23	10107.18	40438.65	80880.36
5	78.96	2527.45	10107.79	40438.48	80877.43
6	78.94	2527.65	10107.00	40438.71	80878.13
7	78.95	2528.36	10107.42	40438.26	80880.73
8	79.12	2527.93	10106.78	40438.94	80883.68
9	78.98	2526.46	10106.68	40438.82	80877.36
10	78.98	2527.58	10106.36	40438.34	80882.59
11	78.96	2526.97	10106.27	40438.11	80879.48
12	78.96	2527.40	10110.58	40438.41	80876.11
13	79.00	2527.92	10110.07	40438.67	80877.60
14	78.95	2527.51	10110.87	40438.89	80880.26
15	79.10	2528.27	10110.67	40438.42	80882.32
16	78.96	2527.94	10110.72	40438.65	80883.53
17	79.01	2527.61	10107.49	40438.74	80880.35
18	78.96	2526.44	10108.16	40438.35	80879.56
19	79.13	2527.76	10111.22	40438.42	80880.52
20	79.15	2526.37	10108.59	40438.61	80881.88
Average	79.00	2527.40	10108.57	40438.52	80879.86
1 device consumed	0.07715	0.07713	0.07712	0.07713	0.07713

In test C, it took approximately 10.84, 66.32, and 523.03 ms to generate 2^{10}, 2^{17}, and 2^{20} ephemeral shared keys, respectively. These results indicate that when SIDF is configured using a GPU, the time taken for SIDF to perform ECIES decryption operation is 1.8 and 0.5 sec, respectively, when roughly 1,000,000 UEs attempt simultaneous access. Figure 11 shows the average decryption speed and average operation time of the corresponding result as a graph.

8.3. Analysis of Results. Several analyses can be performed on the experimental results. First, observe that test A, an experiment that does not perform parallel processing using a CPU, requires 0.07 ms to perform one ECIES decryption operation. However, in tests B and C that perform parallel programming using a GPU, the execution time of one ECIES decryption operation was linearly reduced to that of ephemeral shared keys. After ephemeral shared keys, the operation time was constant at approximately 0.0017 ms and 0.0005 ms. Second, based on the minimum consumption time, notice that the parallelized tests B and C using a GPU are at least 43 and up to 150 times faster than test A, a nonparallelized test using a single CPU. Particularly, when device C is used with numerous CUDA cores, when configuring one SIDF per UE or more, the SIDF configuration efficiency using a GPU can improve by 150 times compared to that achieved when configuring using a single CPU. Third, if SIDF is configured through a GPU with device B specifications, up to 2^{19} UEs can be processed within 1 second. As mentioned earlier, this is crucial when configuring the mMTC 5G network.

The mMTC 5G network requirement states that the network should be configured to provide smooth access when 1,000,000 UEs per square kilometer access the network. Notice that our results satisfy the mMTC 5G network requirement. Therefore, even if it is not a URLLC 5G network requirement, our proposed approach can satisfy the mMTC requirement if a GPU is used. Test B and test C were conducted using the same CUDA source code. Analysing only the results of the two experiments, test C's operating time in ECIES decryption is 3 times faster than test B's operating time. At this time, the number of CUDA cores of device C used at this time is 3 times larger than the number of CUDA cores of device B used in test B. Therefore, it can be seen that the tests using GPU have a speed difference proportional to the number of CUDA cores. Device B and device C used in this experiment are not the latest graphics cards. GPUs released in 2021 have 2.6 times more CUDA cores than device C and are significantly ahead of memory clocks. Therefore, it is expected that better results can be obtained than the results of this experiment when the experiment is performed using the latest GPU. Therefore, if GPU having a specification of device B or higher is used alone or

Table 5: Results of test B (ms).

Iterate count	Number of ephemeral shared keys				
	2^{10}	2^{15}	2^{17}	2^{19}	2^{20}
1	8.46	62.27	229.19	891.86	1756.89
2	8.19	61.71	227.29	893.98	1779.35
3	8.58	61.87	228.89	884.21	1781.72
4	8.23	62.66	225.74	890.49	1778.26
5	9.34	62.17	226.25	886.25	1797.18
6	8.24	61.56	230.06	887.2	1777.28
7	8.17	61.53	227.7	898.44	1781.34
8	8.41	62.4	227.06	885.77	1783.97
9	8.32	61.92	231.07	889.06	1778.21
10	9.04	63.61	227.98	885.33	1783.23
11	8.42	62.14	225.56	898.17	1817.36
12	8.23	62.19	224.99	893.3	1790.2
13	8.4	61.37	226.87	882.89	1806.54
14	8.15	61.46	228.26	890.03	1797.49
15	8.54	61.7	228.1	888.63	1780.03
16	8.15	61.57	227.79	882.48	1797.53
17	8.18	61.92	230.64	897.27	1798.12
18	8.37	62.62	228.52	890.18	1799.07
19	8.23	62.16	228.67	890.06	1777.16
20	9.11	61.71	228.67	892.14	1801.67
Average	8.438	62.02	227.965	889.887	1788.13
1 device consumed	0.00824	0.001893	0.001739	0.001697	0.001705

multiple GPU is used simultaneously using a scalable link interface technique, 5G SIDF function that satisfies the mMTC requirement is configured.

Lastly, we would like to analyze the warp unit operation, which is a disadvantage of using GPU. In the case of a small request of 2^{10} or less, the difference between device A and device B is approximately 10 times. In the case of 1 warp unit, about 2.4 seconds is consumed by the CPU, and if this is calculated again, it can be calculated that device A consumes about 0.07 seconds when processing a single request. Similarly, in the case of device B, about 0.26 seconds is consumed by the use of GPUs in the 5G home network. Previously, the disadvantage of GPU when the calculation is not performed in warp units is that even if the calculation is terminated, the memory and computing device cannot be used until all warps are finished. However, looking at the above results, it can be said that it is faster to use the GPU unconditionally when the number of decryption requests is divided into warp units, and decryption is performed sequentially when the last remaining number is 4 or more. In other words, it is concluded that it is faster to use a general CPU only when there are a maximum of 3 decryption requests, and it is better to use a GPU in all other cases. In particular, this paper is characterized by the use of GPUs in the 5G home network. Previously, in [18–22], GPU was used to quickly operate only ECC cryptographic operations. However, in this paper, the 5G SIDF configuration is divided into three stages, and the GPU is used in all processes. In particular, in stage 1, HN's private key is divided into warp units so that the ECIES key matching process can be performed quickly in the subsequent stage. In addition, in stage 3 of decrypting the actual SUCI into SUPI, a general and effective method using a GPU device in cryptography was used. Therefore, the most important point in this paper is that the GPU is used for all operations performed by SIDF.

In addition, as suggested in this paper, when cryptographic algorithms and cryptographic systems are newly installed in the system, they should be evaluated through various evaluation methods by comparing operation time, power consumption, flexibility, financial cost, etc., with other commercial equipment. This is because these evaluation methods are items to consider when an encryption algorithm or system is operated in actual equipment. Firstly, I would like to compare it with other 5G commercial products in terms of power consumption. Power consumption of cryptographic algorithms becomes particularly important in IoT equipment, embedded equipment, and 5G UE that are currently used in various environments. IoT equipment, embedded equipment, 5G UE, etc., are equipment in which equipment is operated through batteries, not in an environment in which power is generally supplied at all times. Therefore, power consumption when a cryptographic algorithm or cryptographic system operation is added to the device is a very critical issue.

Table 6: Results of test C (ms).

Iterate count	Number of ephemeral shared keys				
	2^{10}	2^{15}	2^{17}	2^{19}	2^{20}
1	11.27	19.69	66.1	262.72	518.48
2	11.24	19.12	66.58	264.77	518.96
3	11.24	18.37	66.71	262.61	521.15
4	11.24	18.67	66.2	263.47	521.29
5	11.24	18.30	66.05	264.77	518.90
6	11.24	18.41	66.09	263.54	525.26
7	11.24	18.13	65.81	264.13	524.52
8	11.24	18.28	65.69	263.81	521.95
9	11.24	18.58	66.65	263.82	523.49
10	11.24	18.15	66.32	263.42	525.21
11	11.24	18.15	65.67	263.82	522.53
12	11.24	18.21	66.04	264.18	522.01
13	10.91	18.60	66.4	263.66	525.47
14	10.13	18.50	65.93	262.78	525.26
15	10.13	18.32	65.77	262.49	523.02
16	10.12	18.81	66.87	264.30	522.45
17	10.13	18.35	67.12	262.10	523.5
18	10.13	18.60	67.26	262.07	523.62
19	10.13	18.28	66.67	260.84	525.54
20	10.13	18.35	66.56	262.27	528.03
Average	10.84	18.49	66.32	263.28	523.03
1 device consumed	0.01058	0.00056	0.00050	0.00050	0.000499

However, the 5G core network environment that this paper focuses on does not require low-power computation and low-power cryptographic algorithms. 5G core network is an environment that requires more high security level (depending cryptographic algorithm key length) and compatibility of cryptographic algorithms than the advantages obtained by using low-power computation and low-power cryptographic algorithms [41]. In addition, the maximum power consumption of device B and device C used in the SIDF configuration method using GPU presented in this paper is about 260 watts [42, 43]. This value is not an absolute comparison because it is the maximum consumption of the GPU device, not the power consumption of the cryptographic algorithm, but it does not show a big difference when compared with the maximum power consumption of 5G equipment.

The second is a comparison of operation time compared to 5G commercial security product solutions. Most 5G network devices collaborate with the world's leading cryptographic equipment manufacturers such as Thales and IDQ to provide security and encryption functions. If we look at the cryptographic devices in these network devices, they focus on many security functions. The cryptographic function used in Ericssons's CCSM is provided through Thales's 5G Luna Hardware Security Module (HSM). All crypto operations and storing, generating, and managing of encryption keys are performed within the secure confines of the 5G Luna HSM FIPS 140-2 level 3 and Common Criteria EAL 4+, while ensuring the protection of subscriber identities, including the Subscription Concealed Identifier (SUPI), user equipment, radio area networks (RANs), and their core network infrastructure. 5G Luna HSM offers up to 1,660 transactions per second (tps) for profile A Decrypt 25519 with a single HSM. In case of High Availability Cluster 2 5G Luna HSMs rather than single HSM, it supports up to 3,440 tps. The 5G Luna HSM offers high assurance key protection and up to 6,070 tps for profile B Decrypt P-256 and 1,660 tps for profile A Decrypt 25519 to meet security and throughput and scalability requirements for 5G [44]. Tables 7 shows the comparison values between test C and Thales 5G Luna HSM tested in this paper.

Also, unlike Ericsson, Nokia's SDM solution does not mention the cryptographic algorithm. However, to configure a key distribution system within the 5G network through collaboration with SKT and IDQ, various security functions such as quantum key distribution (QKD) technology have been added to build a 5G network security system. However, we could not find specific information about 5G primary authentication like SUCI deconcealing mentioned in this paper. Even though the 5G core network is built through collaboration with each mobile operator and equipment manufacturer, the functions that can perform primary UE authentication when trying to access many devices at once in an mMTC environment are still lacking. According to the information disclosed by Ericsson, when ECIES profile A is selected, the performance is 1,660 tps. Of course, since the numerical values

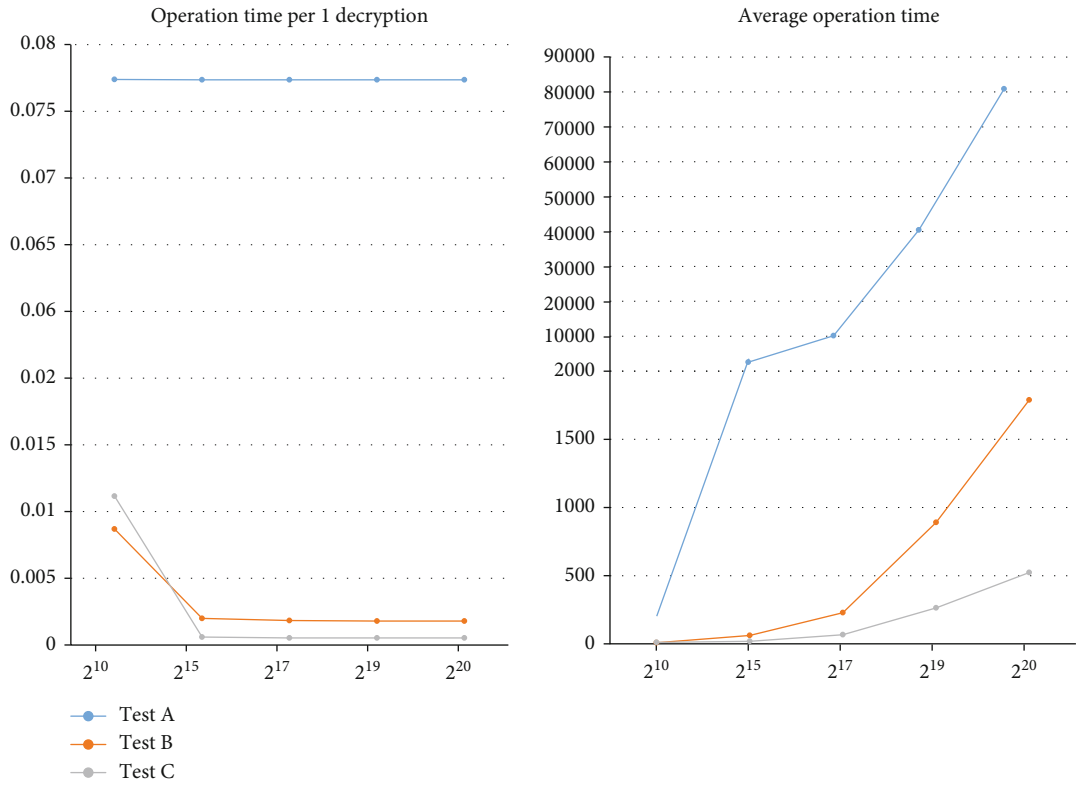

Figure 11: Operation time: each graph means the time required for each number of operations. The graph on the left means one decryption time consumed in each experiment. The graph on the right means the total decryption time consumed in each experiment.

Table 7: Comparison with 5G commercial product.

	Test C	Thales 5G Luna HSM
Power consumption	260 watts (GPU max power consumption)	84~100 watts
Operation time per 1,000,000 UE processing (profile A)	0.523 seconds	290 seconds (with clustering HSM)
Financial cost	Under $1,500	Expensive
Features	—	FIPS 140-2 and CC EAL4+

presented by the equipment are measured based on network transactions, it is difficult to directly compare them with this thesis, which only performs ECIES decryption calculations. However, considering the requirement to cover 1,000,000 devices per square kilometer, which is the standard of mMTC, it can be expected that it will consume about 631 seconds. However, when the method presented in this paper is performed on device C, the ECIES decryption processing time for the same 1,000,000 ECIES is about 0.523 seconds, which is approximately 1200 times faster. The third is financial cost comparison. In order to satisfy the mMTC standard through the aforementioned commercial products, AUSF/SIDF can be configured through multiple devices instead of using one device. However, as suggested in this paper, when a GPU is used, the figure is up to 1000 times faster, and the network configuration is much cheaper in terms of price, so I think the price competitiveness is better than using the existing solution.

9. Conclusion

In 5G telecommunication networks, SUCI is created and used with an ECIES scheme to prevent IMSI information for user identification from being exposed during the initial access of a mobile communication terminal. However, because the ECIES scheme includes a public-key operation for key sharing, significant overhead may occur in the 5G network when many terminals attempt simultaneous access, such as the mMTC service for IoT environment, owing to a large amount of required computation. We judged that it is insufficient to support the current mMTC service for IoT environment through surveys on 5G products and several 5G network open-source surveys. Therefore, in this study, to solve this problem, a parallelization technique that uses a GPU is proposed to configure the SIDF responsible for SUCI deconcealing. In our experiments, a minimal source modification was applied to operate the OpenSSL source code on the GPU, and no separate optimization was

performed. Nevertheless, as the number of device access requests increases, the key sharing time per device when using a GPU is up to 150 times faster than that achieved with a CPU. In addition, the method presented in this paper showed superiority through a comparison of power consumption and operation time with 5G products. Therefore, the experiment presented here proves that the parallel implementation method using GPUs for SIDF configuration can be a countermeasure against the overhead that occurs when numerous devices request access to multiple terminals.

In current experiments, a method of dividing the home network private key used for ECIES according to threads was used. Because GPU resource optimization was not applied separately here, we will experiment with an optimization implementation suitable for the GPU environment in future research. In addition, because this study only tested profile A, a study on the parallelization of profile B needs to be conducted. We plan to conduct research applying the experimental results obtained here to 5G open-source projects.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the Republic of Korea's MSIT (Ministry of Science and ICT), under the High-Potential Individuals Global Training Program (2021-0-01516) supervised by the IITP (Institute of Information and Communications Technology Planning & Evaluation).

References

[1] Series M, *IMT Vision–Framework and Overall Objectives of the Future Development of IMT for 2020 and Beyond*, ITR-U, 2015.

[2] S. P. Rao, S. Holtmanns, I. Oliver, and T. Aura, "Unblocking stolen mobile devices using SS7-MAP vulnerabilities: exploiting the relationship between IMEI and IMSI for EIR access," in *2015 IEEE Trustcom/BigDataSE/ISPA*, pp. 1171–1176, Helsinki, Finland, 2015.

[3] M. Khan, A. Ahmed, and A. R. Cheema, "Vulnerabilities of UMTS access domain security architecture," in *2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pp. 350–355, Phuket, Thailand, 2008.

[4] 3rd Generation Partnership Project (3GPP), *Technical Specification Group Services and System Aspects; TS 33.501: Security Architecture and Procedures for 5G System (Release 16)*, 3GPP, 2020.

[5] 3rd Generation Partnership Project (3GPP), *Technical Specification Group Services and System Aspects; TS 23.503: Policy and Charging Control Framework for the 5G System (5GS); Stage 2 (Release 16)*, 3GPP, 2020.

[6] 3rd Generation Partnership Project (3GPP), *Technical Specification Group Core Network and Terminals; TS 23.003: Numbering, Addressing and Identification; (Release 16)*, 3GPP, 2020.

[7] SECG SEC 1, *Recommended Elliptic Curve Cryptography, Version 2.0*, 2009, http://www.secg.org/sec1-v2.pdf.

[8] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[9] Miller VS, "Uses of elliptic curves in cryptography," in *Advances in Cryptography-Crypto'85*, vol. 218, pp. 417–426, Springer, 1986.

[10] IETF RFC 7748, *Elliptic Curves for Security*, 2016, https://tools.ietf.org/html/rfc7748.

[11] SECG SEC 2, *Recommended Elliptic Curve Domain Parameters, Version 2.0*, 2010, https://www.secg.org/sec2-v2.pdf.

[12] NVIDIA, "Developer forum," https://forums.developer.nvidia.com/.

[13] NVIDIA, "CUDA C++ programming guide 11.3; see," 2007, https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.

[14] Wouter de Groot, *A Performance Study of X25519 on Cortex-M3 and M4, [Ph.D. Thesis]*, Eindhoven University of Technology, 2015.

[15] M. Rivain, "Fast and regular algorithms for scalar multiplication over elliptic curves," 2011, IACR Cryptology ePrint Archive 2011/338, http://eprint.iacr.org/2011/338.pdf.

[16] T. Kudithi and R. Sakthivel, "High-performance ECC processor architecture design for IoT security applications," *The Journal of Supercomputing*, vol. 75, no. 1, pp. 447–474, 2019.

[17] M. Imran, I. Shafi, A. R. Jafri, and M. Rashid, "Hardware design and implementation of ECC based crypto processor for low-area-applications on FPGA," in *2017 International Conference on Open Source Systems & Technologies (ICOSST)*, pp. 54–59, Lahore, Pakistan, 2017.

[18] R. Szerwinski and T. Güneysu, "Exploiting the Power of GPUs for Asymmetric Cryptography," in *Cryptographic Hardware and Embedded Systems – CHES 2008*, Springer, 2008.

[19] D. J. Bernstein, T. R. Chen, C. M. Cheng, T. Lange, and B. Y. Yang, "ECM on graphics cards," in *Advances in Cryptology - EUROCRYPT 2009*, vol. 5479, pp. 483–501, 2000.

[20] S. C. Seo, T. H. Kim, and S. K. Hong, "Accelerating elliptic curve scalar multiplication over GF(2m) on graphic hardware," *Journal of Parallel Distributed Computing*, vol. 75, pp. 152–167, 2015.

[21] J. Dong, F. Zheng, J. Cheng, J. Lin, W. Pan, and Z. Wang, "Towards high-performance X25519/448 key agreement in general purpose GPUs," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, pp. 1–9, Beijing, China, May 2018.

[22] L. Gao, F. Zheng, N. Emmart, J. Dong, J. Lin, and C. C. Weems, "DPF-ECC: accelerating elliptic curve cryptography with floating-point computing power of gpus," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 494–504, New Orleans LA USA, May 2020.

[23] Ericsson, *Cloud Native Subscription and Data Management in 5G – A Guide to Mastering Data and Subscriber's Handling in Multi-Access Core Network*, Technical Paper, 2021.

[24] NOKIA, *Subscriber Data Management - Evolving SDM for 5G, Cloud and Future Networks*, Whitepaper, 2021.

[25] "The open source for 5G," https://open5gs.org.

[26] "The open source for 5G," https://www.free5gc.org/.

[27] "The open source for 5G," https://pkg.go.dev/golang.org/x/crypto/curve25519.

[28] "The open source for 5G," https://openairinterface.org/.

[29] M. Brown, D. Hankerson, J. López, and A. Menezes, "Software implementation of the NIST elliptic curves over prime fields," in *Topics in Cryptology — CT-RSA 2001*, Springer, Berlin, Heidelberg, 2001.

[30] H. Tschofenig and M. Pegourie-Gonnard, "Performance investigations," 2015, http://www.ietf.org/proceedings/92/slides/slides-92-lwig-3.pptx.

[31] A. D. Biagio, A. Barenghi, G. Agosta, and G. Pelosi, "Design of a parallel AES for graphics hardware using the CUDA framework," in *Proc. of 2009 IEEE International Parallel and Distributed Processing Symposium*, pp. 1–8, Rome, Italy, 2009.

[32] S. A. Manavski, "CUDA compatible GPU as an efficient hardware accelerator for AES cryptography," in *2007 IEEE International Conference on Signal Processing and Communications*, pp. 65–68, Dubai, United Arab Emirates, 2007.

[33] K. Iwai, N. Nishikawa, and T. Kurokawa, "Acceleration of AES encryption on CUDA GPU," *International Journal of Networking and Computing*, vol. 2, no. 1, pp. 131–145, 2012.

[34] M. Düll, B. Haase, G. Hinterwälder et al., "High-speed curve25519 on 8-bit 16-bit and 32-bit microcontrollers," *Designs, Codes and Cryptography*, vol. 77, no. 2-3, pp. 493–514, 2015.

[35] E. Mahe and J.-M. Chauvet, "Fast GPGPU-based elliptic curve scalar multiplication," *IACR Cryptology ePrint Archive*, vol. 2014, 2014.

[36] C. Huth, R. Guillaume, P. Duplys, K. Velmurugan, and T. Güneysu, "On the energy cost of channel based key agreement," in *Proceedings of the 6th International Workshop on Trustworthy Embedded Devices*, pp. 31–41, Vienna Austria, 2016.

[37] "The open source toolkit for SSL/TLS, OpenSSL-1.1.1.g," https://www.openssl.org/.

[38] T. Chou, "Sandy2x: "new Curve25519 speed records"," in *Selected Areas in Cryptography – SAC 2015*, O. Dunkelman and L. Keliher, Eds., vol. 9566, pp. 145–160, Springer, Cham, 2016.

[39] H. Fujii and D. F. Aranha, "Curve25519 for the Cortex-M4 and beyond," in *Progress in Cryptology – LATINCRYPT 2017*, pp. 36-37, Springer, 2017.

[40] E. Käsper, "Fast elliptic curve cryptography in OpenSSL," in *Financial Cryptography and Data Security*, vol. 7126, pp. 27–39, Springer, 2011.

[41] N. Mouha, "The design space of lightweight cryptography," 2015, https://hal.inria.fr/hal-01241013.

[42] NVIDIA, "Product specification," https://www.nvidia.com/en-in/geforce/products/10series/geforce-gtx-1060/.

[43] NVIDIA, "Product specification," https://www.nvidia.com/en-us/titan/titan-xp/.

[44] Thales, "Thales 5G Luna network HSM," 2021, https://cpl.thalesgroup.com/resources/encryption/5g-luna-network-hsm-data-sheet.