Solving the Cross Domain Problem with Functional Encryption

Alan Kaminsky
Department of Computer Science
Rochester Institute of Technology
ark@cs.rit.edu

Michael Kurdziel
L3Harris Technologies
Mike.Kurdziel@L3Harris.com

Steve Farris
L3Harris Technologies
Steve.Farris@L3Harris.com

Marcin Łukowiak
Department of Computer Engineering
Rochester Institute of Technology
mxleec@rit.edu

Stanisław Radziszowski
Department of Computer Science
Rochester Institute of Technology
spr@cs.rit.edu

***Abstract*—A Cross Domain Problem (CDP) is the question of how to securely access and exchange information between the domains of varying security levels. A Cross Domain Solution (CDS) addresses the CDP by designing the framework and protocols for such access and transfers. Most existing CDS methods rely on policies and trusted parties to manage different security levels. A CDS that can function in the presence of untrusted parties is a challenge.**

Functional Encryption (FE) is an encryption scheme in which a secret key allows one to compute a specific function of plaintext from the ciphertext. FE is a generalization of identity-based and attribute-based encryption frameworks. General and simultaneously practical FE is an emerging area, and only special types of encryption schemes and functions are effectively handled within existing systems.

We apply the concepts of FE to explore a new solution to the CDP, and we argue that our solution does not leak information, provided that widely accepted assumptions about standard digital signatures hold. We built a practical software case study application using a trusted Key Distribution Center (KDC), a standard symmetric key block cipher component (like the AES), and using the Elliptic Curve Digital Signature Algorithm (ECDSA). The experiments show that the computational overhead introduced to routing by our method is cost effective, where the additional cost is equivalent to just a few applications of standard digital signatures.

***Index Terms*—Functional Encryption, Cross Domain Problem, Cross Domain Solution**

I. Introduction

The Cross Domain Problem (CDP) refers to the need for systems and processes which allow the transfer of information that is classified at different security levels to be securely communicated between domains at different classification levels [1]. Most current solutions are based on a system of risk management that usually relies on trusted actors. A fully automated CDS is desired which allows an information source to transfer data across a heterogenous network of security domains, in a secure manner, to an appropriate destination. End to end encryption offers a potential solution. However, a method is required that allows encrypted traffic to be routed over a network to an end destination without revealing sensitive information. For example, revealing source and destination IP addresses will make the system vulnerable to traffic flow analysis. An adversary could also use the addresses to spoof the information source or destination. One potential solution is Content-Based Routing (CBR) [2]. A Content-Based Router examines the message content and routes the message to a different channel or destination based on data contained in the message. However, CBR cannot be used as a basis for a CDS because gateways and routers in different domains cannot have access to the data. Homomorphic Encryption (HE) schemes offer another potential solution to the CDP. HE schemes allow computations to be performed on encrypted data without the need to decrypt it. This could allow untrusted or semi-trusted routers and gateways to determine limited information to operate on the encrypted data without first decrypting the data. A CDS was proposed in [3] that was based on the Yet Another Somewhat-Homomorphic Encryption (YASHE) scheme [4]. This CDS was prototyped and proved feasible, however practical limitations such as high data expansion, high latency and significant processing requirements were observed.

Functional Encryption (FE) is a generalization of public-key encryption where a secret key enables the user to evaluate a single function on a piece of encrypted data [5]. Application of FE to the CDP offers a potentially limited but more feasible solution. The objective of the investigation described in this paper was to explore and assess the viability of using functional encryption to create a CDS that ensures security while transferring information across different security domains with routing decisions determined by untrusted or semi-trusted gateways and routers.

This paper is organized as follows: The CDP and the past approaches to it are reviewed in Section II. Our proposed FE scheme for a CDS is described in Section III. The case study describing the framework and its implementation for achieving a CDS using FE are presented in Section IV and Section V. The experimentation results and conclusions are provided in Section VI and Section VII, respectively.

II. Background

A. The Cross Domain Problem

Multiple data sources produce data of varying classifications that need to be routed to specific endpoints. The encrypted data is multi-casted onto an untrusted network to multiple network untrusted gateways. The gateways will process the data and determine only if the data is intended for its network endpoints. If true, then the gateway will forward the encrypted data. If not true, the gateway will drop the data assuming it is intended for another gateway. No gateway will learn anything about the type of data, the classification of the actual endpoint or the ultimate destination of data. This prevents a rogue gateway and network from searching for specific information and gathering data not intended for its endpoint. The system can include multiple types of classification, such as: Top Secret, Secret, Confidential, Restricted, Official, or Unclassified. An endpoint network may have one or more associated users. A figure that represents the concept of the problem is shown in Fig. 1 from [1]. This figure illustrates a system whereby each data source encrypts the data it produces with a secret symmetric key that is only shared between the data source and the ultimate end point. Next, the data source concatenates an attribute which identifies the subnetwork of the endpoint. The attribute is bound to the encrypted data packet by signing the entire packet with a private "concealing key" associated with a specific gateway. That gateway will hold a corresponding public "revealing key". Then the data source multicasts the packet using a library of IP addresses to all the gateways that sit on the network. Each gateway will use its "revealing key" to check the attribute of the incoming packet. If the gateway is able to verify the attribute, then it broadcasts the packet to its endpoints, otherwise it drops the packet. Only the endpoint that shares the symmetric key with the data source will be able to decrypt the payload of the packet. Note that the revealing key only permits a gateway to determine that a packet is intended for one of its endpoints. It is not able to determine any other gateway that the packet might be intended for. The gateway also cannot access the plaintext data. As such, the gateway is considered to be partially or semi-trusted.

In summary, a successful CDS must guarantee its resistance against unauthorized gateway's action and that the intruders cannot learn anything, namely:

- Gateway cannot:
  - Determine plaintext (lacks symmetric key)
  - Determine any attribute value which is not its own (lacks revealing keys)
  - Forge or alter ciphertext or concealed attributes (lacks symmetric key and concealing keys)
  - Deduce concealing key (digital signature property)
- Intruder cannot:
  - Determine plaintext (lacks symmetric key)
  - Determine any attribute value (lacks revealing keys)
  - Forge or alter ciphertext or concealed attributes (lacks symmetric key and concealing keys)

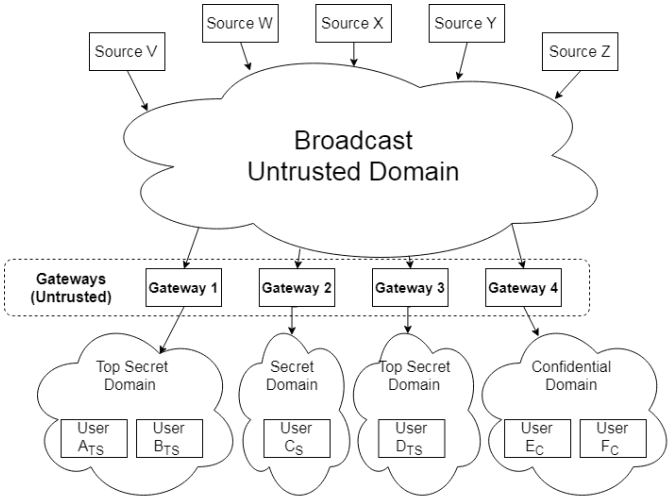

Fig. 1: The Cross Domain Network Case Study [1]

The difficulty of designing a CDS is that, in order to operate on the plaintext data, any party on the network must be protected and trusted [1]. The National Institute of Standards and Technology (NIST) provides a large catalog of security frameworks. These include a number of non-mutually exclusive scenarios for CDS, in the information flow enforcement section, from which one can select to fulfill their requirements [6]. However, the methods and policies used are primarily determined by the implementing organization. Many such CDS related definitions focus on security policy and risk management using protected domains with authorized human personnel to manually evaluate and control the flow of information. Therefore, a system that is able to automatically use information while maintaining security inside an untrusted domain is a challenging problem.

The use of homomorphic or functional encryption to achieve this objective is a mostly unexplored research area. In our previous work we proposed and confirmed as feasible a solution based on homomorphic encryption using YASHE for HE and SIMON-encrypted headers of data blocks being transferred [3]. The overhead of this solution, however, was very substantial. In this work we present a simple yet appealing solution using functional encryption.

B. Functional Encryption

Functional Encryption (FE) is an encryption scheme in which a secret function key allows one to compute a specific function of plaintext from the ciphertext (see Fig. 2). FE is a generalization of identity-based and attribute-based encryption frameworks [5].

General but also practical FE is an emerging area, and only special types of encryption schemes and functions are effectively handled within existing systems. The general solutions to FE schemes tend to be computationally expensive, and thus not widely adopted. Here, however, we use FE to conceal only a few bit-attributes, and the FE restricted this way can be

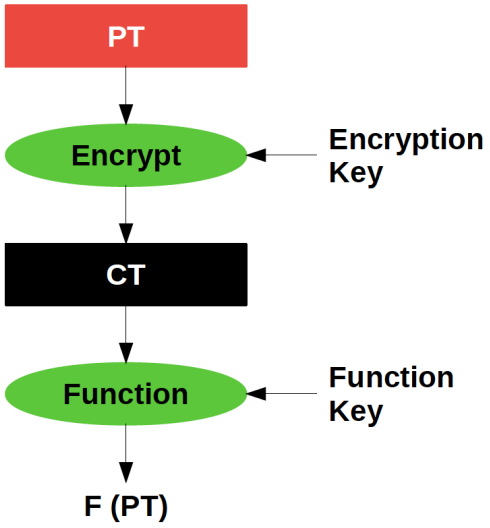

Fig. 2: Starting Functional Encryption

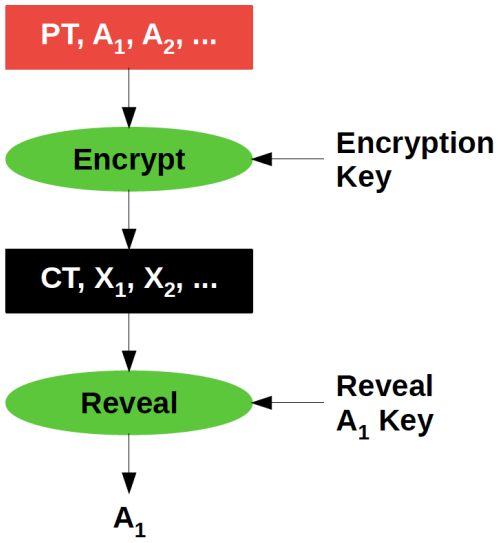

Fig. 3: Using Functional Encryption

effectively implemented with just somewhat adjusted classical mechanism borrowed from digital signatures.

III. Proposed FE-CDS Solution

A. The Setting

Parties known as senders generate sensitive messages (plaintexts PT). A sender encrypts a plaintext with a secret encryption key K using a symmetric cipher, which yields a ciphertext CT. The sender attaches one or more Boolean-valued attributes A_i to the ciphertext and conceals the attribute values as described below in part B, The Protocol.

These senders are data sources depicted in Fig. 1. The key K is used with a block cipher like the Advanced Encryption Standard (AES). Such keys need to be generated and properly shared between the data sources and users prior to initiating our proposed CDS, for each sender/user pair which will need to route some contents in the CDS. The generation, distribution and use of these symmetric keys is not discussed further in this paper, though the Key Distribution Center (KDC) outlined in the *Key Setup* paragraph in the following part B may be used for their generation and distribution. Other means than KDC are possible, like special secured communication channels used by sources and users for key sharing.

The concealed attributes are as follows:

- PT, sensitive content, plaintext,
- A_1, A_2, ..., Boolean attributes,
- CT, encrypted content, ciphertext,
- X_1, X_2, ..., concealed attributes A_i.

Parties known as receivers receive ciphertexts along with their attached concealed attributes (see Fig. 3). A particular receiver is authorized to learn the true/false value of a particular attribute or attributes. A receiver cannot learn the true/false value of any attribute for which the receiver is not authorized. A receiver cannot (correctly) attach any concealed attributes to any ciphertexts. A receiver cannot learn any plaintext.

Parties known as intruders want to break the system's security. An intruder cannot learn any plaintext. An intruder cannot (correctly) attach any concealed attributes to any ciphertexts. An intruder cannot learn the true/false values of any attributes. If an intruder alters a ciphertext and/or its concealed attributes, no one will be able to learn the attribute values; this is a denial-of-service attack, but it does not break the system's security.

B. The Protocol

The protocol relies on a digital signature algorithm. For concreteness, let this be the Elliptic Curve Digital Signature Algorithm (ECDSA) defined in FIPS PUB 186-4 Digital Signature Standard (DSS) [7] with Curve P-192.

Key Setup. A trusted Key Distribution Center (KDC) generates a signature key pair for each attribute A_i. The key pair consists of a signing key SK_i and a verifying key VK_i. Both the signing key and the verifying key are kept secret; the verifying key is not made public as is usually the case. The KDC securely sends each attribute's signing key to each sender. The KDC securely sends each attribute's verifying key to each receiver that is authorized to learn the true/false value of that attribute.

Sender. A sender generates a ciphertext C, determines the value of each attribute A_i, and generates each concealed attribute X_i as follows: If the value of A_i is false, then $X_i = \texttt{Sign}(C\|0, SK_i)$, otherwise $X_i = \texttt{Sign}(C\|1, SK_i)$.

The sender attaches all the concealed attributes X_i to the ciphertext C and sends the message. With ECDSA/P-192, each concealed attribute (signature) occupies 384 bits (48 bytes).

Receiver. A receiver obtains a ciphertext C along with its attached concealed attributes X_i. The receiver is authorized to learn the true/false value of a particular attribute A_j; thus, the

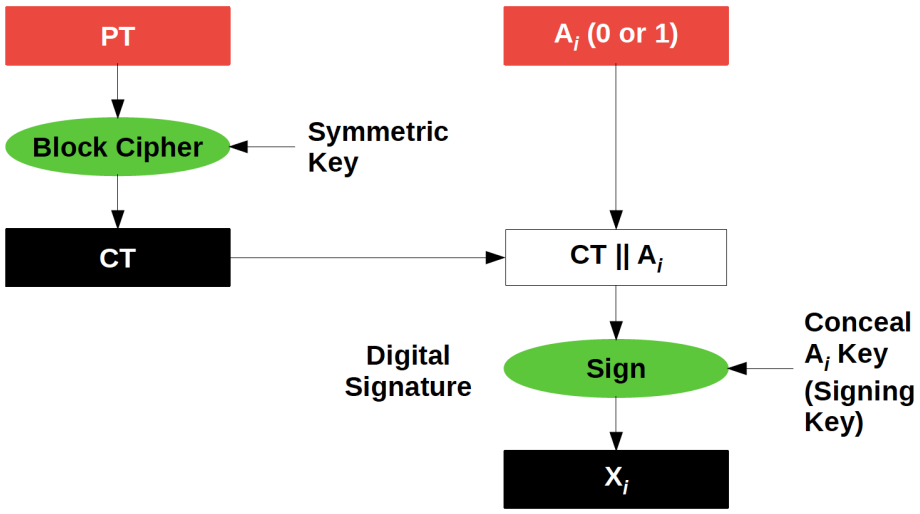

Fig. 4: Attribute Concealment
Concealed attribute is bound to its ciphertext; signature ECDSA with NIST Curve P-192, X_i is 384 bits (48 bytes)

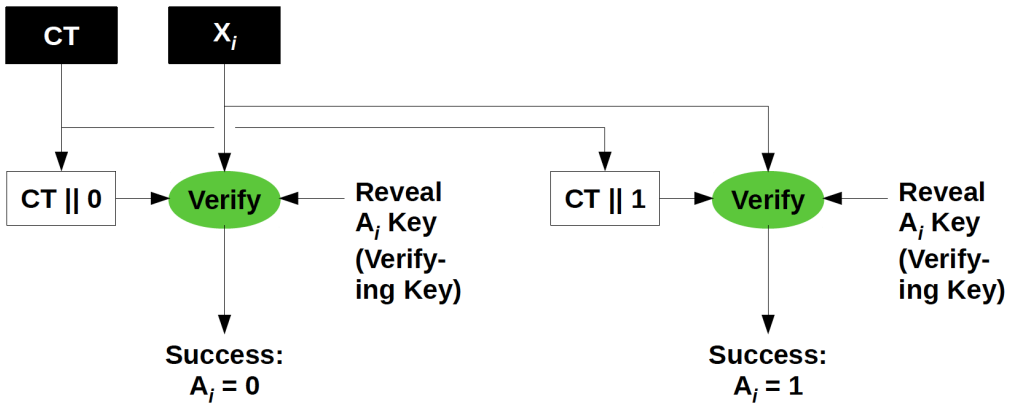

Fig. 5: Attribute Revelation
If both verifications fail, then CT, X_i, and/or reveal A_i key is invalid

receiver knows the verifying key VK_j. For the receiver the following holds: If $\texttt{Verify}(C\|0, X_j, VK_j)$ succeeds, then A_j is false; if $\texttt{Verify}(C\|1, X_j, VK_j)$ succeeds, then A_j is true; otherwise we have an error of unknown value of A_j.

The details of attribute concealment and revelation are illustrated in Fig. 4 and Fig. 5, respectively.

Security. Because no receiver or adversary has the symmetric key used to generate the ciphertexts, no receiver or adversary can learn the plaintexts. If a receiver (or adversary) is not authorized to learn the true/false value of A_i, then that party will not have VK_i, and that party will not be able to carry out the above computation. If that party nonetheless tries to learn the attribute's value using the wrong verifying key, both signature verifications for X_i will fail. Because no receiver or adversary has the signing key SK_i for any attribute, no receiver or adversary can generate a correct concealed value X_i for any attribute. Also, it is not possible to derive the signing key SK_i from the verifying key VK_i. If a party tries to generate X_i using the wrong signing key, then both signature verifications for X_i will fail. If an adversary alters a ciphertext C or a concealed attribute X_i, then both signature verifications for X_i will also fail. Because ECDSA includes a random ephemeral value in every signature computation, the concealed attribute value X_i will be different in every message, even if the ciphertext C and attribute value A_i are the same. Thus, no party will be able to correlate the concealed attribute values with the actual attribute values.

For the signature scheme we use, we have to keep both the signing and verifying keys secret. Thus, it is important that an adversary cannot recover the verification key from the message being signed, $C\|0$ or $C\|1$, and the signature X_i.

In some signature schemes, this is an important aspect of the public verification key. In fact, some of the standards provide recommendations to prevent the verifying key from

being fully or partially recoverable. This is the case for the ECDSA, for which partial recovery of the curve point Q is possible given the signed message, the message signature and other parameters of the elliptic curve. Here $Q = kP$, P is a known curve generator recommended in the standard and integer k is secret. For the case of the RSA, often the public key is recommended to be a small number of the form 2^m+1, such as 3 or 17.

For our work, we use the ECDSA and recommend that the verifying key consist of all curve parameters, including the number of points and the generator P. If a specific curve such as P-192 is desired, we can simply replace its standard generator P with a point mP for a randomly chosen integer m. If we use the RSA signature scheme, then the verifying key should be chosen randomly from the full set of possible keys (an integer co-prime to the modulus).

IV. Use Case: Sensitive Document Distribution

The ciphertexts are encrypted sensitive documents. There are four Boolean attributes: A_1 - document may be sent on an *Unclassified* network, A_2 - document may be sent on a *Secret* network, A_3 - document may be sent on a *Top Secret* network, and A_4 - document may be sent on a *Ludicrously Secret* network. The senders are devices that create documents (ciphertexts). A sender that creates a document at a particular secrecy level attaches concealed attributes to the ciphertext as follows: for Unclassified documents all attributes are true, i.e. $A_1 = A_2 = A_3 = A_4 = 1$, for Secret documents $A_1 = 0$ and $A_2 = A_3 = A_4 = 1$, for Top Secret document $A_1 = A_2 = 0$ and $A_3 = A_4 = 1$, and for Ludicrously Secret document $A_1 = A_2 = A_3 = 0$ and $A_4 = 1$. The first receivers are gateway devices for networks. Each network has a secrecy level. A network's gateway decides whether a particular document is allowed to be sent on a particular network. The gateway learns the true/false value of the document's attribute associated with the network's secrecy level, and allows the document to be sent on the network only if the attribute is true.

The proposed CDS scheme can be extended to attributes with N possible discrete values ($N > 2$). To create a concealed attribute, sign $C\|0, C\|1, C\|2, \ldots$ and/or $C\|N–1$ as appropriate. Determining the actual attribute value from the concealed attribute value takes longer, as up to N signature verifications have to be performed.

V. CDS Framework Using FE

Fig. 6 shows the test/demonstration setup for a Cross Domain Solution based on the proposed Functional Encryption scheme. As previously stated, this approach offers a solution for specific fully automated use cases, but has the advantage of being computationally more feasible than HE based methods. Prior to beginning processing and data transfer, a set of keys will need to be distributed by the Key Distribution Center (KDC). First, a set of encryption keys, shown as ek_0, ek_1 and ek_2 in the figure, are generated and distributed to the data producer and to the intended end points behind Gateway 0, 1 and 2, respectively. These keys allow the data producer to secure the payload so that it is only accessible by authorized endpoints. Secondly, a set of "signing keys", shown as sk_0, sk_1 and sk_2 in the figure are generated and distributed to the data producer. These keys allow the data producer to cryptographically bind gateway attributes to the data payload. Finally, a set of "verification keys", shown as vk_0, vk_1 and vk_2 in the figure are generated and distributed to Gateway 0, 1 and 2, respectively. To process and transfer information, the data producer first encrypts the data payload using the encryption key for the endpoint in the destination domain. Then the producer appends the set of attributes for the gateway to that domain and cryptographically binds the attributes to the payload by using the appropriate signing key. Finally, the data producer multi-casts the secured data item to all of the network gateways. When each gateway receives the secured data item, it can use its verification key to determine if that item is intended for an endpoint in its domain or not. Depending on the result of the verification operation, the gateway will either pass the data item into its domain or it will discard the item. Each gateway is only capable of determining that a secured data item is intended for its domain. It cannot determine the identity of any other destination domain and it cannot access any aspect of the data item itself. In this way, the gateway is considered as an untrusted or partially trusted resource.

VI. Results

For functional verification of the approach, we utilized four Python scripts that were executed on five Linux workstations with Internet network connections. These scripts were designed to mimic the roles of the Key Distribution Center (KDC), Producer, network Gateways, and destination Domains. Python *ecdsa* and *cryptography.hazmat* packages provided support for key generation and cryptographic operations. Socket programming was used to send and receive messages between workstations.

Timing measurements were gathered on a PC with AMD Ryzen 5 3600X 6-Core Processor 3.80 GHz, 16.0 GB RAM 2133 MHz, PCIe 4.0 SSD. A set of small images was used as PT data and the number of attributes selected for this test was four. Nonce (8 bytes), AAD (23 bytes of additional authenticated data) and plaintext image data (127 kB) were concatenated and encrypted using AES with GCM (Galois Counter Mode) and 128-bit secret key. Signing and verification of the attributes was accomplished using the Elliptic Curve Digital Signature Algorithm (ECDSA) over NIST Curve P-192. Sample times are presented in Table I.

	Time [μs]		
Computation	Producer	Gateway	Domain
AES-GCM encrypt	222		
Signing attributes	1929		
Verifying attributes		5568	
AES-GCM decrypt			521

TABLE I: Sample test times

This method has negligible processing time and memory overhead compared with our previous HE-CDS work in [3].

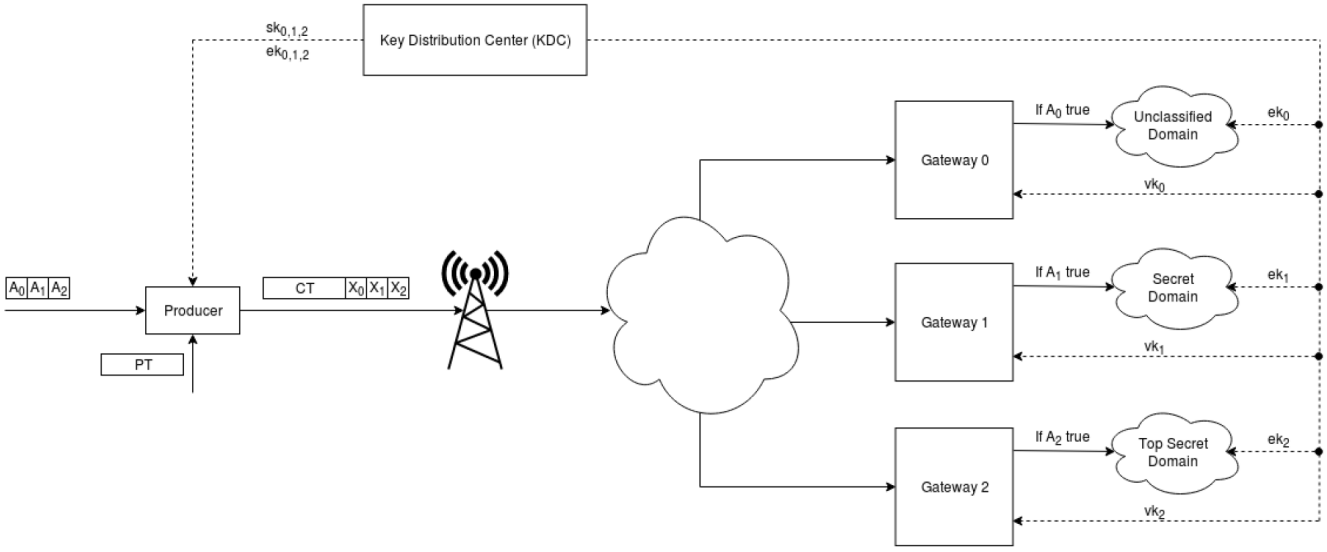

Fig. 6: Setup of a test system

Table I shows that this method can provide a CDS with 128 bits of security while requiring approximately 2 ms to process the data for transmission and 6 ms to evaluate the data. In addition, memory requirements for all the computations are negligible. In contrast, for the same level of security, the HE-CDS based method in [3] requires approximately 35 minutes to process the data for transmission and 20 hours to evaluate the data. In addition, the HE-CDS computations require a total of over 3.5 GB of memory.

VII. Conclusions

This work demonstrated that a CDS can be achieved using functional encryption. We demonstrated that a practical application that securely transfers cross domain information across an untrusted network can be achieved under a parameter selection that ensures 128-bit security using AES and ECDSA. This method requires attributes to be defined a priori and is therefore less flexible than the HE-CDS based method presented in [3]. However, the computational requirements are modest, and the performance is greatly improved.

Table I shows that this method can provide a CDS with 128 bits of security while requiring approximately 2 ms to process the data for transmission and 6 ms to evaluate the data. In addition, memory requirements for all the computations are negligible. In contrast, for the same level of security, the HE-CDS based method in [3] requires approximately 35 minutes to process the data for transmission and 20 hours to evaluate the data. In addition, the HE-CDS computations require a total of over 3.5 GB of memory.

References

[1] Committee on National Security Systems. Committee on National Security Systems (CNSS) Glossary. (4009):160, 2015.

[2] Pedro Bizarro, Shivnath Babu, David DeWitt, and Jennifer Widom. Content-based routing: Different plans for different data. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway*, pages 757–768, Sep 2005.

[3] C. Tinker, K. Millar, A. Kaminsky, M. Kurdziel, M. Lukowiak, and S. Radziszowski. Exploring the application of homomorphic encryption to a cross domain solution. In *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, pages 1–6, 2019.

[4] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. Cryptology ePrint Archive, Report 2013/075, 2013. https://eprint.iacr.org/2013/075.

[5] Dan Boneh, Amit Sahai, and Brent Waters. Functional Encryption: Definitions and Challenges. Proceedings of Theory of Cryptography Conference (TCC), 2011.

[6] NIST. Security and Privacy Controls for Federal Information Systems and Organizations Security and Privacy Controls for Federal Information Systems and Organizations. *Sp-800-53Ar4*, pages 400+, 2014.

[7] NIST Federal Information Processing Standards Publication. Digital Signature Standard (DSS). *FIPS PUB 186-4*, pages 130+, 2013.