In this paper how to use distributed servers is of fundamental sources over the Internet.

\[ p = 1, \ldots, p \]

Customer’s data security, problem of existence of \( p \) frequently changed.

\[ p \]

Due to possible breaches, both accidental and deliberate.

Cloud computing relies on sharing of resources to achieve coherence and economies of scale, similar to a utility (like the electricity grid) over a network. The goal of cloud computing is to allow users to take benefit from all of these technologies, without the need for deep knowledge about or expertise with each one of them. Cloud computing security processes should address the security controls the cloud provider will incorporate to maintain the customer’s data security, privacy and compliance with necessary regulations.

Cloud Computing presents an added level of risk because essential services are often outsourced to a third party, which makes it harder to maintain data security and privacy, support data and service availability, and demonstrate compliance.

Security concerns relate to risk areas such as external data storage, dependency on the “public” internet, lack of control, multi-tenancy and integration with internal security.

The approach to securing Data in the cloud which are stored on distributed servers and back it up on a single server and allow access upon the use of passwords that are needed to be frequently changed.

In this paper we discuss the key distribution system for securing data in cloud.

**Galois field (Finite Field) \((GF(2^p))\)**

A finite field is also called a Galois Field. It is so named in honor of Évariste Galois, a French mathematician. Galois is the first one who established the following fundamental theorem on the existence of finite fields:

**Fundamental theorem of existence of finite field:**

An order-\( n \) finite field exists if and only if \( n = p^m \) for some prime \( p \) (\( p \) is called the characteristic of this finite field) and some positive integer \( m \). We usually use \((GF(p^n))\) to represent the finite field of order \( p^n \).

Galois field is also known as finite field that contains a finite number of elements. Galois fields are important in number theory, cryptography; coding theory etc. It is particularly useful in translating computer data as they are represented in binary forms.

The elements of Galois field is defined as,

\[
GF(p^n) = (0, 1, 2, \ldots, p - 1) \cup (p, p + 1, p + 2, \ldots, p + p - 1) \\
\cup (p^2, p^2 + 1, p^2 + 2, \ldots, p^2 + p - 1)
\]
\[ U \ldots \cup \left( p^{n-1}, p^{n-1} + 1, p^{n-1} + 2, \ldots, p^{n-1} + p - 1 \right). \]

Where \( p \) is any prime number and \( n \) is positive integer.

Finite fields of order \( 2^n \) are called binary fields or characteristic-two finite fields. They are of special interest because they are particularly efficient for implementation in hardware, or on a binary computer. In binary system we represent each value with 0 and 1. Ultimately, binary system offers an alternative way of representing the elements of Galois field. [3] So a bit is an element of \( \text{GF}(2) \). Also byte which is equivalent to 8 bits is an element of \( \text{GF}(2^8) \).

In our research paper we will discuss about computing so we are focusing on Galois field of order 2 and \( 2^8 \).

**Arithmetic of Galois field \( \text{GF}(2^n) \)**

**Addition and subtraction:**

A Galois field \( \text{GF}(2^n) \) is consist of polynomials of degree \( n - 1 \) or less, and their coefficients are 1 or 0. Let \( f \) and \( g \) are two polynomials belonging to \( \text{GF}(2^n) \), with coefficients \( a_{n-1}, a_{n-2}, \ldots, a_1, a_0 \) and \( b_{n-1}, b_{n-2}, \ldots, b_1, b_0 \) then addition of \( f \) and \( g \) is defined as,

\[ h = f + g \]

If \( c_{n-1}, c_{n-2}, \ldots, c_1, c_0 \) are coefficient of polynomial \( h \) then,

\[ c_m = a_m + b_m \mod 2 \]

Where \( 0 \leq m \leq n - 1 \).

Similarly subtraction of \( f \) and \( g \) is defined as,

\[ h = f - g \]

and if \( c_{n-1}, c_{n-2}, \ldots, c_1, c_0 \) are coefficient of polynomial \( h \) then,

\[ c_m = a_m - b_m \mod 2 \]

Where \( 0 \leq m \leq n - 1 \).

**Multiplication and multiplicative inverse**

Let \( \Theta(2) \) be an irreducible polynomial of degree at least \( n \) in \( \text{GF}(2^n) \), irreducible means it cannot be factored into two or more polynomials in \( \text{GF}(2^n) \) and each of the degree less than \( n \). Then multiplication of \( f \) and \( g \) is defined as,

\[ h = (F, g \mod \Theta(2)) \]

The multiplicative inverse of \( f \) is given by \( \Xi(2) \) such that

\[ (F, \Xi(2)) \mod m(2) = 1. \]

Construction of irreducible polynomial:

We now that there is one irreducible polynomial required for multiplication and multiplicative inverse. Thus the problem of how we can generate irreducible polynomials of a given degree.

Theorem 1.1: There exists a probabilistic algorithm which, given as input a finite field \( \mathbb{F}_q \) and positive integer \( n \), produces as output an irreducible polynomial \( p(x) \in \mathbb{F}_q [x] \) of degree \( n \) using Order \((n^4 \log (q))\) operations.

The following lemma gives an explicit formula for the exact number of irreducible monic polynomials over \( \mathbb{F}_q \) of degree \( n \).

**Lemma 1.1:** The number \( N_q(n) \) of monic irreducible polynomials of degree \( n \) in \( \mathbb{F}_q [x] \) is given by:

\[ N_q(n) = \frac{1}{n} \sum_{d|n} \mu \left( \frac{n}{d} \right) q^d = \frac{1}{n} \sum_{d|n} \mu(d) q^n \]

Where \( \mu \) is the Mobius function

\[ \mu: N \to \{-1, 0, 1\} \]

defined as,

\[ \begin{align*}
\mu(n) &= 1 & \text{if } n = 1 \\
\mu(n) &= (-1)^k \text{ if } n \text{ is product of } k \text{ distinct prime } \\\n\mu(n) &= 0 & \text{if } n \text{ is divisible by square of prime }
\end{align*} \]

**Key distribution in cloud computing**

Galois field widely used in cryptography since each data are represented as a vector in finite field encryption and decryption using mathematical arithmetic is very straightforward and easy. [4] In cloud computing mostly used data partitioning scheme in which data are stored in various system on cloud but when size of data are very large then it is inefficient to create data partition and distribute over the network. As shown in figure 1, data are partitioned and stored in different system.

Figure 1: Data partition method


[424-427]
For this type of problem user may wish to encrypt the data and store on the trusted single server and keep encryption key secret. The encryption key is almost very large and can not easy to remember it, therefore user may create partition of key and distribute them over the different system on the network. The key distribution of public keys is done through public key servers as shown in figure 2. When a person creates a key-pair, he keeps one key private and the other, public-key, is uploaded to a server where it can be accessed by anyone to send the user a private, encrypted, message.

Figure 2: Key Partition Method

Key partitioning using Galois field \(GF(2^n)\)

We use polynomials in Galois field \(GF(2^n)\), for key partitioning which is better than polynomial generalization of power encryption transformation. A Galois field \(GF(2^n)\) consist of polynomials of degree \(n-1\) or less, such that their coefficients lie in \(GF(2)\).

Let \(a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 x + a_0\), where \(a_i = 1\) or \(0\). be a polynomial in \(GF(2^n)\). It may written in binary representation as \(a_{n-1}a_{n-2} \ldots a_1a_0\). Now we generate \(p\) number of random polynomials of any random degree in \(2^n\), say \(k_1(x), k_2(x), \ldots, k_p(x)\).

Let \(\Theta(x) = x^n + d_{n-1}x^{n-1} + \ldots + d_1 x + d_0\), where \(d_j \in GF(2^n)\) be any irreducible polynomial.

Now we perform multiplication of product of \(p\) -polynomials modulo \(\Theta(x)\) as defined earlier.

\(k_1(x)k_2(x) \ldots k_p(x) \equiv \chi(x) \mod \Theta(x)\)

Where \(\chi(x)\) is the key polynomial.

\(\chi(x) \equiv k_1(x)k_2(x) \ldots k_p(x) \mod \Theta(x)\)

The random key generate is binary representation of \(\chi(x)\) say, and the key partitions are binary representation of polynomials \(k_1(x)k_2(x) \ldots k_p(x)\) say \(k_{1,1}k_{2,1} \ldots k_{1,p}\).

References


