Original Research Article

EdDSA over Galois Field GF(p^m) for multimedia data

ABSTRACT

The Edwards-curve Digital Signature Algorithm (EdDSA) was proposed to perform fast public-key digital signatures and thus replace the Elliptic-Curve Digital Signature Algorithm. Its key advantages over the latter include higher performance and straightforward, secure implementation for embedded devices. EdDSA algorithm is implemented over Galois Field. The operations like addition and multiplication in Galois field are different compared to normal addition and multiplication. Hence implementing EdDSA over Galois field provides more security compared to the conventional EdDSA signature. The basics of Galois Field and its application to store data is introduced. The finite field GF (p^m) is an indispensable mathematical tool for some research fields such as information coding, cryptology, theory and application of network coding.

Key Words- EdDSA, ECDSA, Galois Field, Authentication

1. INTRODUCTION

The digital signature is a digital code which is computed and authenticated by a public key encryption (like RSA, EcDSA) and is then attached to an electronically transmitted document for verification of its contents and also the identity of the sender.

The Edwards-curve Digital Signature Algorithm is a variant of Schnorr's signature system with Edwards's curves that may possibly be twisted. The public-key signature algorithm EdDSA is similar to ECDSA which was proposed by Bernstein. EdDSA is defined for two twisted Edwards's curves which are edwards25519 and edwards448. EdDSA needs to be instantiated with certain parameters. Creation of signature is deterministic in EdDSA and it has higher security due to intractability of some discrete logarithm problems. Thus, it is safer than DSA and also ECDSA which require high quality randomness for each and every signature computed.

Generally, a point P= (x, y) lies on E, a twisted Edwards curve if it verifies the following formula: $ax^2 + y^2 = 1 + dx^2y^2$ where a, d are two distinct, non-zero elements of the field M over which E is defined. Itis untwisted in the special case where a=1, because the curve reduces to an ordinary Edwards curve. Consider 'a' and 'd' values in the above equation as 10 and 6 respectively. The equation becomes → $10x^2+y^2=1+6x^2y^2$. For which the plot of Edward curves is shown below:

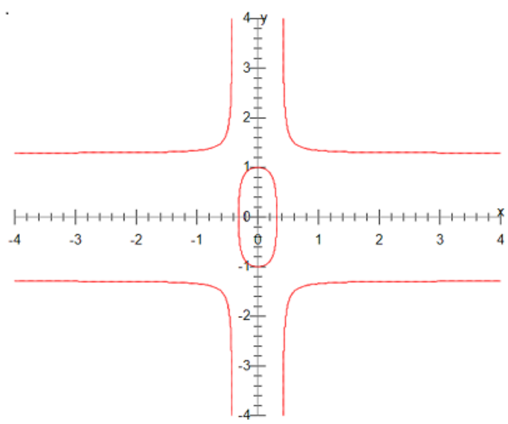


Fig.1. plot of Edward curves for a=10 and d=6

Galois Field, named after Evariste Galois, also known as finite field, refers to a field in which there exist a finite number of elements. It is very useful in translating computer data so that they are represented in binary forms.

That is, computer data which is limited to a combination of two numbers, 0 and 1are the components in Galois field with number of elements being two. By representing data as a vector in a Galois Field, we can easily perform scrambled mathematical operations. The elements of Galois Field GF (p^m) can be defined as:

GF(p^m) = (0,1,2,...,p−1) ∪ (p,p+1,p+2,...,p+p−1) ∪ (p^2,p^2+1,p^2+2,...,p^2+p−1) ∪...............∪ (p^{m-1},p^{m-1}+1, p^{m-1}+2,...,p^{m-1}+p−1)

Where p∈P and m∈Z^+. The order of the field is given by p^m while p is called the characteristic of the field. GF refers to Galois Field. Also, the degree of polynomial of each element is at most m−1. For example GF(4) = (0, 1, 2, 3) which consists of 4elements in which each of them is a polynomial of degree ‘0’ (a constant) while GF(2^4) = (0, 1, 2, 3*…..15*)and consists of 2^4= 16elements where each of them is a polynomial of degree at most 2. GF (2^4) defines the basic arithmetic operations over the finite set of bytes.

2. METHODOLOGY

2.1 KEY PAIR GENERATION PROCESS

For the EdDSA authenticator to function, it needs to know its own private key. The public key is obtained from the private key and the parameters specified over domain. The private key is not accessible to any third party. The public key must be openly read accessible. The public and private key pair ensures data is protected during transmission.

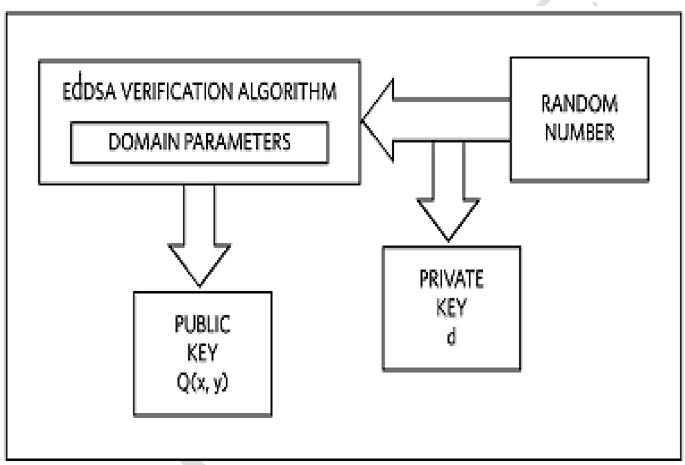
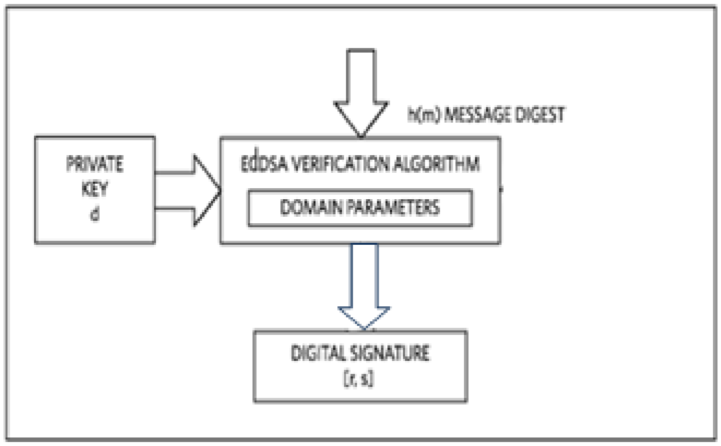


Fig.2. For generation of private and public keys

2.2 SIGNATURE COMPUTATION PROCESS

The digital signature allows the receiver of a message to verify the message's authenticity using the sender's public key. It also offers non-repudiation, that is, the source of the message cannot deny the validity of the data sent.



`

Fig.3. Generation of Digital Signature

2.3 SIGNATURE VERIFICATION PROCESS

The signature verification is the counterpart of the signature computation. Its purpose is to verify the message's authenticity using the authenticator's public key. In order to reduce frauds, signature verification is very important. It increases accuracy and efficiency.

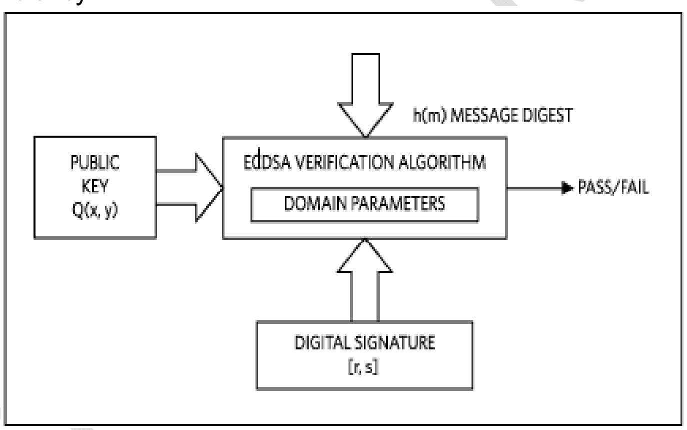


Fig.4. Verification of Digital Signature

2.4 EDWARDS-CURVE DIGITAL SIGNATURE ALGORITHM

EdDSA is a public-key signature algorithm similar to ECDSA proposed by Bernstein et al. [2]. In the paper RFC 8032 [4], EdDSA has been defined for two twisted Edwards curves edwards25519 and edwards448; but, the EdDSA can also be instantiated over other curves. Generally speaking, a point P =(x, y) lies on E, a twisted Edwards curve if it verifies the following formula: $ax^2 + y^2 = 1 + dx^2y^2$ where a, d are two distinct, non-zero elements of the field K over which E is defined.

In practice the public key and the signatures are output according to the encoding defined in RFC 8032. Since there is a one-to-one relation between curve elements and encoded values we do not detail the encoding in our description. The signature (R, S) of a message M is computed according to following algorithm[1].

2.4.1 EdDSA SIGNATURE ALGORITHM

Requires: Message (M), hash digest values of message → (h_0, h_1,………, h_{2b-1}), Private key (B) and Public key (A) derived from B

1: a ← $2^{b-2} + \sum_{3 \le i \le b-3} 2^i h_i$
2: h ← H (h_b,....., h_{2b-1}, M)
3: r ← h mod GF(*l*)
4: R ← r · B
5: h ← H (R, A, M)
6: S ← (r + ah) mod GF(*l*)
7: return (R, S)

EdDSA uses a private key that is b-bit long and a hash function H that produces a 2b - bit output. One common instance is to use SHA-512 for b = 256 bits[1].
In this paper, Hash algorithm used is SHA-1.Length of Hash, of 2b bit length is 20 bits. So, private key is of 10 bits.
An integer 'a' is determined from H(k) = (h_0,h_1,...,h_{2b-1}) with **"a = $2^{b-2} + \sum_{3 \le i \le b-3} 2^i h_i$".** The public key A is then computed from the base point B ≠ (0 , 1) of order *l*, chosen as per the EdDSA specifications [2], such that A = a·B. Where *l* can be any abstract algebraic equation, for example, *l* = $a^2 + b^2$,which is defined over Galois field p^m. '***p***' is a positive prime number raised to value '***m***'. For this algorithm, using Galois field for the calculation of value ***'l'*** gives more security and reduces computational time.
Even if multiple signatures are computed for an identical message, same signature is obtained. Thus, EdDSA is deterministic in nature.

2.4.2 VERIFICATION OF EDDSA SIGNATURE

A signature is considered valid if it satisfies the following equation, 8S·B mod *l* = (8·R + 8H (R, A, M) ·A) mod *l*. Verification without the cofactor 8 is a stronger way to verify a signature. Since the algorithm has good performance, ease of implementation, small key and signature sizes, it is rapidly being adopted in security of embedded devices also.

2.5 SOFTWARE REQUIREMENTS

MATLAB version 9.5 [R2018b].
Toolboxes required: Symbolic math toolbox and Communications Toolbox.

3. RESULTS

Computed signature for text data is obtained as below.

Case 1: The private key given as input at receiver end matches with key set by sender of the message. Thus, signature is verified as being correct.

```
MATLAB Command Window

The message is
today is sunday
The hash values for the given message are
   159    34   107    89    84   105   130   251   232   211   163    40    96   161↙
166   112   236   154   215   153

 Enter the private key:***
 the public key is: 57168
R =

  1×20 uint64 row vector


   184   126   212   158    94    58   216    86   108     0   148   196    42    14↙
104    34    10   152   114    56
S =      1×20 uint64 row vector

    26    74    43   109     9    14    53    54    97    20    48    91    69    84↙
50   115   121    71    57    67


the signature is
  Columns 1 through 20

   184   126   212   158    94    58   216    86   108     0   148   196    42    14↙
104    34    10   152   114    56
  Columns 21 through 40
    26    74    43   109     9    14    53    54    97    20    48    91    69    84↙
50   115   121    71    57    67
Signature is correct
```

Fig.5. Results for the correct signature

Case 2: The private key given as input at receiver end does not match with key set by sender of the message. Thus, signature is incorrect.

```
The message is
today is sunday
The hash values for the given message are
  159    34   107    89    84   105   130   251   232   211   163    40    96   161↙
166   112   236   154   215   153

 Enter the private key  ***
  the public key is
      1857960
R =  1×20 uint64 row vector

      Columns 1 through 17

  5980   4095   6890   5135   3055   1885   7020   2795   3510      0   4810   6370
1365    455   3380   1105    325

 Columns 18 through 20

 4940   3705   1820

S =   1×20 uint64 row vector
   113    67    26    19    20    66   113    88    62    91    29   117   108    16↙
13   100    49     6   101    98

the signature is
  Columns 1 through 17
   5980   4095   6890   5135   3055   1885   7020   2795   3510      0   4810   6370↙
1365    455   3380   1105    325
Columns 18 through 40

   4940   3705   1820    113    67    26    19    20    66   113    88    62↙
91     29    117    108     16    13    100     49      6    101     98

   signature is not correct
```

Fig.6. Results for incorrect signature.

3.1. DIFFERENCES BETWEEN PROPOSED AND EXISTING SYSTEM

Signature computation is deterministic in EdDSA and its security is based on the intractability of some discrete logarithm problems. Thus it can be concluded that, though key length is very small, it is safer than DSA and ECDSA which require high quality randomness for each and every signature.

In the proposed system, EdDSA is implemented over Galois Field in which the operations are different from normal arithmetic operations. Hence it is more secure than the existing EdDSA system. Confidentiality and integrity for multimedia data like text and image also increases.

Several parameters of ECDSA and proposed EdDSA over Galois Field are compared and tabled as follows:

Table 1. Performance comparison of EdDSA with ECDSA

PARAMETERS	ECDSA	PROPOSED EdDSA OVER GF(p^m)
Key length	384	10
Key generation (sec)	0.799	0.0006
Signature generation(sec)	0.0016	0.0002
Signature verification(sec)	0.0082	0.0007

4. CONCLUSION

Nowadays, digital signatures are being used all over the Internet. Digital signature schemes are also used in electronic transactions over block chain. An algorithm to give a digital signature that authenticates text data and provides non-repudiation is designed. It is faster than existing digital signature schemes and has a relatively small key length. Yet, it does not compromise on the data integrity and security it offers. Implementation of certain calculations over Galois field reduces computation time and size of the signature. This algorithm can be extended to verify integrity of multimedia like image, video, etc.

REFERENCES

[1] YolanRomailler, Sylvain Pelissier, “Practical fault attack against the Ed25519 and EdDSA signature schemes”, In Workshop on Fault Diagnosis and Tolerance in Cryptography, 2017

[2] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In International Workshop on Cryptographic Hardware and Embedded Systems, pages 124–142. Springer, 2011.

[3] Daniel J. Bernstein, Simon Josefsson, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. EdDSA for more curves. Cryptology ePrint Archive, Report 2015/677, 2015.

[4] IlariLiusvaara and Simon Josefsson. Edwards-Curve Digital Signature Algorithm (EdDSA). RFC 8032, January 2017.

[5] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. “Twisted edwards curves”, in Cryptology ePrint Archive, Report 2008/013, 2008.

[6] Niels Duif, Tanja Lange et al, "High-speed high-security signatures”, in Journal of Cryptographic Engineering, 2011.