岐阜大学工学部
電気電子・情報工学科
令和 6 年度卒業論文

セキュアな V2V アドホックネットワーク ルーティングプロトコルのための EdDSA 署名方式の評価

三嶋研究室

学籍番号：1213033107

永野 正剛

指導教員：三嶋 美和子 教授

目次

はじめに

第 1 章　準備

1.1　VANET

vanet 書くよ

1.2　GPSR

GPSR 書くよ

1.3　デジタル署名

デジタル署名書くよ

第 2 章　EdDSA

EdDSA(**Edwards-curve Digital Signature Algorithm**) とは, 2012 年に Daniel J. Bernstein らによって提案された決定論的シュノア署名 [?] の一種であり, 効率的な点の加算公式をもつ特殊な楕円曲線であるエドワーズ曲線, またはツイストエドワーズ曲線 [?] を利用したデジタル署名である.[?] EdDSA の開発は, 従来の ECDSA が持ついくつかの欠点を克服する目的で進められた. 2017 年, インターネット技術やプロトコル, 標準, ソフトウェアなどに関する公式な文書である RFC 8032[?] が, インターネットの技術的な標準を策定する国際的な組織 IETF によって公開された. さらに, 2020 年には国立標準技術研究所 (National Institute of Standards and Technology, NIST) による情報処理標準規格 FIPS 186-5 で標準化された. EdDSA は ECDSA に比べて多くの利点があるため, インターネット上で通信する際にデータを安全に暗号化し, 通信相手の認証を行うためのプロトコルである TLS(Transport Layer Security)1.3 [?] やネットワーク経由で他のコンピュータと安全に通信するためのプロトコルである SSH , ブロックチェーン技術などの様々なプロトコルやアプリケーションで採用されている [?].

RFC8032 によると, EdDSA では暗号攻撃に対して約 128 ビットのセキュリティレベルの *Ed25519* と, 約 224 ビットのセキュリティレベルの *Ed448* の 2 種類の実装のどちらかを安全性要件に応じて利用するよう推奨されている. Ed25519 は 2005 年に Bernstein らによって提案された Curve25519 を基盤としており, 特に「ツイストエドワーズ曲線」という形式を採用することで, 計算効率とセキュリティを向上させている. 現在, Ed25519 は EdDSA の最も一般的なインスタンスであり, 必要なリソースが少なく, 十分な安全性を提供している. よって, 本研究では Ed25519 を用いてデジタル署名を実装する.

この章では Ed25519 について概説する. 2.1 節では Ed25519 で使用されるデータの変換について, 2.2 節では Ed25519 のパラメータについて述べる. 2.3 節では Ed25519 の理解に必要となる楕円曲線とその上での演算について述べる. 2.4 節では Ed25519 のデジタル署名アルゴリズムについて述べ, 最後に, 2.5 節では ECDSA と Ed25519 のセキュリティと計算効率の比較を行い, Ed25519 を用いる利点を説明する.

2.1 データの変換

EdDSA のアルゴリズム内では, 整数や点をオクテット列に変換するエンコードとその逆変換であるデコードが行われる.

以下で使用されるデータの変換について説明する.

オクテット

8 ビットからなるビット列

$$b_0b_1b_2b_3b_4b_5b_6b_7$$

を**オクテット**と呼ぶ. 厳密には 8 ビット以外を指すこともある「バイト」の代わりに, 必ず 8 ビットのことを指すものとして使われている語である.
なお, 最下位ビットは b_0, 最上位ビットは b_7 である.

例. 数値 $0d128$ のオクテットに対応するビット列は 00000001 である.

リトルエンディアン形式

リトルエンディアン形式とは、数値をバイト単位で格納する際に, 最下位バイト（数値の最小の値を持つバイト）を先頭に配置し, 続けてその次に小さいバイトを配置していく方法を指す. これは、通常の十進法で右端から左へ数字を読むのと逆の順番でデータを並べることになる.

例えば、32 ビットの数値 $0x12345678$ をリトルエンディアン形式でメモリに格納すると, 次のような順番でバイトが並ぶ：

- 最下位バイト (1 バイト目)：$0x78$
- 2 バイト目：$0x56$
- 3 バイト目：$0x34$
- 最上位バイト (4 バイト目)：$0x12$

このようにして, $0x12345678$ はメモリ上で $0x78, 0x56, 0x34, 0x12$ の順番に格納される.

エンコードとデコード

1. ENC(s)
   b ビットの整数 s を入力として, s を リトルエンディアン形式にして $\frac{b}{8}$ オクテットに変換して出力する.
2. DEC(t)
   オクテット列 t を入力として, 整数 s に変換して出力する. つまり, $\mathrm{DEC}(t) = \mathrm{ENC}^{-1}(t) = s$ である.
3. ENCE(A)
   ツイストエドワーズ曲線 E 上の点 A(x, y) の y を入力として, ENC(y) によりオクテット列 y' に変換し, y' の最終バイト (最終オクテット) の最上位ビットに x 座標の符号（$x \geq 0$ ならば 0, $x < 0$ ならば 1）を格納したオクテット列を出力する.
4. DECE(t)
   オクテット列 t を入力として, ツイストエドワーズ曲線 E 上の点 (x,y) に変換して出力する.
   (a) t の最終オクテットの最上位ビットを x 座標の符号として取り出し x_0 に格納する. （$x_0 = 0$ または, $x_0 = 1$ とする.）
   (b) t の最終オクテットの最上位ビットを 0 に設定する.
   (c) y =DEC(t) を計算し, $0 \leq y < p$ でないならばデコード失敗.
   (d) 以下の処理を行う.
      i. $u = y^2 - 1$, $v = d * y^2 + 1$ として $x = uv^3(uv^7)^{\frac{p-5}{8}} \bmod p$ を計算する.
      ii. $vx^2 \neq \pm u \bmod p$ ならばデコード失敗.
      iii. $vx^2 = -u \bmod p$ ならば, $x = 2^{\frac{p-1}{4}} x$
   (e) $x = 0$ かつ $x_0 = 1$ ならばデコード失敗.
   (f) x_0 が $x \bmod 2$ と異なるならば $x = p - x$ とする.
   (g) 点 (x, y) を出力する.

2.2 パラメータ

EdDSA のパラメータは以下の通りである.

- p : 法となる素数. EdDSA は $\mathbb{F}_p$ 上の楕円曲線を使用する.
- b : $b \geq 10$ かつ $p < 2^{b-1}$ となる正整数. 公開鍵の長さを表す.
- E' : エンコーディング関数.
- H : ハッシュ関数. 2b ビット長のハッシュ値を出力する.
- (a, d, c, l) : 楕円曲線 E を決定するパラメータ.

$$E \coloneqq \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p \mid ax^2 + y^2 = 1 + dx^2y^2\}$$

– a は $\mathbb{F}_p$ 上の平方剰余. すなわち、$x^2 \equiv d(modp)$ となる x が存在する.

– d は非ゼロの非剰余. すなわち、$x^2 \equiv d(modp)$ となる x が存在しない.

– c はコファクタであり、2 または 3.

– l は 2^{200} より大きい奇素数で E の位数 $\#E = 2^c l$ となるような数であり、B の位数.

- n : $c \leq n < b$ となる整数.
- B : E 上のベースポイント. $B \neq (0, 1)$

コファクタに関しては**参考論文**を参照していただきたい. 本研究で使用する Ed25519 のパラメータは以下の通りである.

- p : $2^{255} - 19$
- b : 256
- E' : 2.2 で述べた Ed25519 のエンコーディング関数
- H : SHA-512
- a : -1
- d : $-\frac{121665}{121666}$
- c : 3
- l : $2^{252} + 27742317777372353535851937790883648493$
- n : 254
- B : (15112221349535400772501151409588531511454012693041857206046113283949847762202, 46316835694926478169428394003475163141307993866256225615783033603165251855960)

2.3 楕円曲線

この節では Ed25519 で使用される**ツイストエドワーズ曲線 (Twisted Edwards Curve)** について説明する.

ツイストエドワーズ曲線は体 $\mathbb{F}_p$ (p は素数かつ $p \neq 2$) 上で定義される. 非零かつ異なる要素 $a, d \in \mathbb{F}_p$ を持つとき、以下の式で与えられる.

$$ax^2 + y^2 = 1 + dx^2y^2 \tag{2.1}$$

楕円加算

点の加算式は以下の通りである.

$$(x_1, y_1) + (x_2, y_2) = (\frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 + ax_1x_2}{1 - dx_1x_2y_1y_2}) \tag{2.2}$$

$\mathcal{O}$ は無限遠点と呼ばれ、E は $\mathcal{O} := (0, 1)$ を単位元とする加法群になる．エドワーズ曲線では、楕円点の加算、2 倍算（同一の点の加算)、無限遠点の加算が同一式で表現される． このため、ECDSA のような分岐処理が不要になり、計算が簡略化される.

2.4 デジタル署名アルゴリズム

Ed25519 における 3 つのアルゴリズムの手順を以下に述べる.

鍵生成

1. 法とする素数 p、楕円曲線 E、基準点 B、鍵のサイズ b、ハッシュ関数 H、エンコーディング関数 E'、コファクター c、位数 L を定める.
2. b バイトのランダムな値 sk を生成し、**秘密鍵**とする.
3. $h = H(sk)$ を計算し、h（オクテット文字列）を前半部分 $h[0]$ から $h[31]$ と後半部分 $h[32]$ から $h[63]$ に分ける.
4. 前半部分の最初のバイト $(h[0])$ の下位 3 ビットを 0 にクリアする． 最後のバイト $(h[31])$ の最上位ビットを 0 に、最上位 2 ビット目を 1 に設定したものをリトルエンディアンの整数として解釈し、スカラー $s \pmod L$ を生成する.
5. 基準点 B を使って $A = sB$ を計算し、$ENCE(A)$ を**公開鍵**とする.

スカラー値 s は 8 の倍数で，正確に 255 ビットとなる

署名生成

1. 秘密鍵 sk を使って、ハッシュ値 $h = H(sk)$ を計算する.
2. h の後半部分 $h[32]$ から $h[63]$ を使って、

$$r = H(h[32] \,||\, ... \,||\, h[63] \,||\, M) \bmod L.$$

を計算し、デコードする.

3. $R = [r]B$ を計算し、エンコードする.
4. $k = H(R \,||\, A \,||\, M) \bmod L$ を計算し、デコードする.
5. 鍵生成の際 s を用いて $S = (r + k * s) \bmod L$ を計算し、エンコードする.
6. (R, S) を署名とする.

署名検証

1. メッセージ M と署名 (R, S) を受け取る.
2. $R' = DECE(R)$ として、R をデコードする.
3. $S' = DEC(S)$ として、S をデコードする.
4. $A' = DECE(A)$ として、A をデコードする.
5. 2〜4 でデコードに失敗した場合や $0 \leq S < L$ の範囲外である場合、棄却する.
6. $k' = H(R' \,||\, A' \,||\, M) \bmod L$ を計算し、デコードする.
7. $[8][S']B = [8]R' + [8][k']A'$ が成り立てば、署名は有効である. そうでなければ、署名は無効である.

2.5 ECDSA と Ed25519 の比較

Ed25519 は ECDSA と比べ、以下の点でセキュリティと効率性が向上している.

セキュリティ

決定論的な署名生成

Ed25519 は、署名ごとにランダムな値を生成する代わりに、メッセージと秘密鍵をハッシュすることでノンス（署名に使用するランダムな数値）を決定論的に生成する. この方法により、乱数生成の失敗による秘密鍵の漏洩リスクを回避している. 一方、ECDSA では、乱数が予測可能または重複した場合、秘密鍵が簡単に推測されるリスクがあり、これがセキュリティの大きな弱点となる.

計算効率

Ed25519 は、ねじれたエドワーズ曲線を使用しており、この形式の曲線は楕円曲線演算を効率的に実行できる特性を持っている． 特に、加算と倍加の操作が簡素化され、計算ステップが少なく済むため、ソフトウェアでの実装が高速になる．

乱数生成の回避

EdDSA では乱数を鍵生成でのみ使用している．この乱数は秘密鍵に直接影響を与えるため，暗号的に安全である必要がある．一方で，署名生成，署名検証では乱数を使用しない．一般的に，暗号的に安全な乱数は生成の時間がランダムで処理コストが高い.EdDSA では署名生成，署名検証を一定時間で行い，乱数を使用するアルゴリズムに比べて高速に処理することができる．

乗法逆元の不要

Ed25519 では拡張した射影座標系（拡張ツイストエドワーズ座標）で計算することがRFC8032 で推奨されている． この座標系の特性により、処理コストの高い乗法逆元の処理が不要となり処理が単純化され、高速化しやすい．

第 3 章　提案手法

第 4 章　シミュレーション環境

第 5 章　シミュレーション実験

第 6 章　EdDSA に関する実装評価まとめ

おわりに

謝辞

参考文献