paozip for Node.js

Node.js ソースコードを暗号化して、大切なコードを守るツール

ユーザーズマニュアル

バージョン 2.0.0

2026年2月

有限会社 パオ・アット・オフィス

https://www.pao.ac/

目次

1. はじめに

- 1.1 paozip for Node.js とは
- 1.2 特長
- 1.3 仕組み（ラッパー方式）
- 1.4 PHP/Python版との違い

2. 動作環境

- 2.1 対応 OS
- 2.2 対応 Node.js バージョン
- 2.3 対応フレームワーク

3. セットアップ

- 3.1 Docker で試す（おすすめ）
- 3.2 npm からインストール
- 3.3 ソースからインストール
- 3.4 インストールの確認

4. クイックスタート

- 4.1 暗号化してみよう
- 4.2 ラッパーを作ってみよう
- 4.3 実行してみよう
- 4.4 paozip run で直接実行

5. CLI コマンドリファレンス

- 5.1 encrypt（暗号化）
- 5.2 decrypt（復号）
- 5.3 run（実行）
- 5.4 check（暗号化チェック）
- 5.5 version（バージョン表示）

6. JavaScript API リファレンス

- 6.1 paozip モジュール
- 6.2 importer モジュール

7. require フックの使い方

- 7.1 基本的な使い方
- 7.2 動作の仕組み
- 7.3 注意事項

8. 暗号化キーについて

8.1 キーの設定方法

8.2 キーファイルの作成

8.3 注意事項

9. 各種環境での利用

9.1 Express

9.2 npm パッケージとして配布

9.3 Docker

10. トラブルシューティング

11. 使用許諾

12. 代金支払い方法

1. はじめに

・1.1 paozip for Node.js とは

・1.2 特長

・1.3 仕組み（ラッパー方式）

・1.4 PHP/Python版との違い

1. 動作環境

・2.1 対応 OS

・2.2 対応 Node.js バージョン

・2.3 対応フレームワーク

1. セットアップ

・3.1 Docker で試す（おすすめ）

・3.2 npm からインストール

・3.3 ソースからインストール

・3.4 インストールの確認

1. クイックスタート

・4.1 暗号化してみよう

・4.2 ラッパーを作ってみよう

・4.3 実行してみよう

・4.4 paozip run で直接実行

1. CLI コマンドリファレンス

・5.1 encrypt（暗号化）

・5.2 decrypt（復号）

・5.3 run（実行）

・5.4 check（暗号化チェック）

・5.5 version（バージョン表示）

1. JavaScript API リファレンス

・6.1 paozip モジュール

・6.2 importer モジュール

1. require フックの使い方

・7.1 基本的な使い方

・7.2 動作の仕組み

・7.3 注意事項
1. 暗号化キーについて
・8.1 キーの設定方法
・8.2 キーファイルの作成
・8.3 注意事項
1. 各種環境での利用
・9.1 Express
・9.2 npm パッケージとして配布
・9.3 Docker
1. トラブルシューティング
1. 使用許諾
1. 代金支払い方法

1. はじめに

1.1 paozip for Node.js とは

「Node.js で書いたコードを、暗号化して配布したい...」

そんな願いを叶えるのが paozip for Node.js です。

あなたの Node.js アプリケーション — Express で構築した API サーバー、社内ツール、SaaS のバックエンド — そのソースコードを暗号化して、安全に配布できます。

しかも、暗号化されたコードはそのまま動きます。require するだけで自動的に復号。利用者はソースコードの中身を見ることなく、アプリケーションを実行できるのです。

1.2 特長

- Pure JavaScript 実装 — ネイティブ依存なし。どの環境でも動きます
- Windows / macOS / Linux 全環境対応
- require フック — 暗号化 .jse ファイルを透過的に require できる
- Express / Fastify 対応 — 普段のフレームワークでそのまま使える
- PHP版・Python版と同じ暗号化エンジン — zencode による堅牢な暗号化
- npm でインストール — いつもの npm install でOK

1.3 仕組み（ラッパー方式）

paozip for Node.js は ラッパー方式（方式A） を採用しています。

```
app.js ─
  │
  ├── const importer = require('paozip/lib/importer')
  ├── importer.install()          ← require
  │
  └── const secret = require('./secret')  ← secret.jse
        │
        └──  (.jse)
```

ポイント：

- app.js（ラッパー）は暗号化しません。これが「入口」になります
- secret.jse（暗号化ファイル）は require するだけで自動復号
- 利用者からは、普通に Node.js を実行しているのと変わりません

1.4 PHP/Python版との違い

項目	PHP版	Python版	Node.js版
暗号化方式	同一（zencode）	同一（zencode）	同一（zencode）
実行方式	透過実行	ラッパー方式	ラッパー方式
C 拡張	あり（.so）	あり（.so）	なし（Pure JS）
run コマンド	なし	あり	あり
ファイル拡張子	.php	.pye	.jse

Node.js版はPython版と同じラッパー方式ですが、Pure JavaScript で実装されているため、ネイティブモジュールのビルドは不要です。

2. 動作環境

2.1 対応 OS

OS	備考
Windows 10/11	公式 Node.js インストーラー
macOS 12+	nvm / 公式インストーラー
Ubuntu 20.04+	apt / nvm / n
CentOS/RHEL 8+	dnf / nvm
その他 Linux	Node.js 18+ が動作すれば OK

2.2 対応 Node.js バージョン

バージョン	対応状況
Node.js 18 LTS	OK
Node.js 20 LTS	OK（推奨）
Node.js 22 LTS	OK

> Node.js 16 以前は非対応です。

2.3 対応フレームワーク

フレームワーク	対応状況
Express	OK
Fastify	OK
Koa	OK
CLI スクリプト	OK

3. セットアップ

3.1 Docker で試す（おすすめ）

まずは Docker で手軽に試してみましょう。面倒なインストールは一切不要です。

```
cd paozip-nodejs-demo
docker-compose build
docker-compose run --rm paozip-demo
```

コンテナの中に入ったら：

```
#
cat lib/secret.js              # ←

#
cat lib/secret.jse             # ←

#
node app.js                    # ←

# paozip run
paozip run lib/secret.jse      # ← OK
```

暗号化前は丸見えだったコードが、暗号化後はまったく読めない。でも動く。 — これが paozip の魅力です。

3.2 npm からインストール

```
#
npm install paozip

# CLI
npm install -g paozip
```

Pure JavaScript なので、ネイティブモジュールのビルドは不要です。

3.3 ソースからインストール

納品された paozip-nodejs ディレクトリを使って：

```
cd paozip-nodejs
npm install
npm link
```

3.4 インストールの確認

```
# CLI
paozip version
# → paozip for Node.js 2.0.0 ...

# API
node -e "const p = require('paozip'); console.log(p.getVersionString())"
```

4. クイックスタート

3分で暗号化を体験しましょう。

4.1 暗号化してみよう

まず、暗号化キーを作成します：

```
echo "a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6" > .paozip_key
```

次に、秘密のコードを作ります：

```
// secret_logic.js
module.exports = {
  calculate(x) {
    //
    return (x * 31337 + 42) % 65536;
  },

  apiKey() {
    return "sk-very-secret-key-12345";
  }
};
```

暗号化！

```
paozip encrypt secret_logic.js
#  secret_logic.js → secret_logic.jse

#
paozip check secret_logic.jse
# → secret_logic.jse: Encrypted (paozip format)
```

4.2 ラッパーを作ってみよう

```
// app.js —
const importer = require('paozip/lib/importer');
importer.install();

//
const secret = require('./secret_logic');

//
console.log(secret.calculate(100));
console.log(secret.apiKey());
```

4.3 実行してみよう

```
#  .js .jse
rm secret_logic.js

#
node app.js
# → 4272
# → sk-very-secret-key-12345
```

暗号化されたファイルだけで、ちゃんと動きました。

4.4 paozip run で直接実行

ラッパーを書かずに、暗号化ファイルを直接実行することもできます：

```
paozip run secret_logic.jse
```

5. CLI コマンドリファレンス

5.1 encrypt（暗号化）

```
paozip encrypt <file.js> [-o output]
```

オプション	説明
<file.js>	暗号化する JavaScript ファイル
-o output	出力ファイルパス（省略時: .jse に変換）

例：

```
paozip encrypt lib/secret.js
# → lib/secret.jse

paozip encrypt lib/secret.js -o encrypted/secret.jse
```

5.2 decrypt（復号）

```
paozip decrypt <file.jse> [-o output]
```

オプション	説明
<file.jse>	復号する暗号化ファイル
-o output	出力ファイルパス（省略時: .js に変換）

例：

```
paozip decrypt lib/secret.jse
# → lib/secret.js
```

> 復号には暗号化時と同じキーが必要です。

5.3 run（実行）

```
paozip run <file.jse> [args...]
```

オプション	説明
<file.jse>	実行する暗号化ファイル
args...	スクリプトに渡す引数

例：

```
paozip run script.jse
paozip run script.jse --port 3000 --env production
```

5.4 check（暗号化チェック）

```
paozip check <file>
```

例：

```
paozip check lib/secret.jse
# → lib/secret.jse: Encrypted (paozip format)

paozip check lib/plain.js
# → lib/plain.js: Not encrypted (plain text)
```

5.5 version（バージョン表示）

```
paozip version
# → paozip for Node.js 2.0.0 [TRIAL]
```

6. JavaScript API リファレンス

6.1 paozip モジュール

```
const paozip = require('paozip');
```

メソッド	説明	戻り値
encrypt(data, key?)	Buffer を暗号化	Buffer
decrypt(data, key?)	Buffer を復号	Buffer
encryptString(str, key?)	文字列を暗号化	Buffer
decryptString(data, key?)	復号して文字列に	string
encryptFile(path, output?, key?)	ファイルを暗号化	string（出力パス）
decryptFile(path, key?)	ファイルを復号	Buffer
decryptFileTo(path, output?, key?)	ファイルを復号して保存	string（出力パス）
isEncrypted(data)	暗号化チェック	boolean
isEncryptedFile(path)	ファイルの暗号化チェック	boolean
isLicensed()	ライセンス版かどうか	boolean
getLicenseEmail()	ライセンスメール	string
getProductInfo()	製品情報	Object
getVersionString()	バージョン文字列	string

6.2 importer モジュール

```
const importer = require('paozip/lib/importer');
```

メソッド	説明
install()	require フックをインストール
uninstall()	require フックをアンインストール
isInstalled()	フックがインストール済みかどうか

7. require フックの使い方

paozip for Node.js の「キラー機能」です。暗号化ファイルを、普通の require と同じ感覚で読み込めます。

7.1 基本的な使い方

```
// app.js
const importer = require('paozip/lib/importer');
importer.install();    // ← 1

//  require
const secret = require('./secret');  // secret.jse
```

たったこれだけ。install() を呼ぶだけで、Node.js の require.extensions に paozip のハンドラが登録されます。

7.2 動作の仕組み

1. require('./secret') が呼ばれる
2. Node.js 標準の require が secret.js を探す
3. 見つからない場合、paozip ハンドラが secret.jse を探す
4. secret.jse が見つかれば、復号してモジュールとして評価

> .js と .jse の両方が存在する場合、.js が優先されます。暗号化後は必ず元の .js ファイルを削除してください。

7.3 注意事項

・importer.install() はアプリケーションの起動時に 一度だけ 呼び出してください
・ES Modules（import 構文）との組み合わせでは、createRequire を使ってください
・暗号化キーは環境変数または .paozip_key ファイルで事前に設定してください

8. 暗号化キーについて

8.1 キーの設定方法

暗号化キーは以下の優先順序で検索されます：

1. API 呼び出し時に直接指定 — paozip.encrypt(data, "my_key")
2. 環境変数 — PAOZIP_KEY
3. キーファイル — .paozip_key（カレントディレクトリから親ディレクトリを順に検索）

8.2 キーファイルの作成

```
echo "a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6" > .paozip_key
```

環境変数で設定する場合：

```
# Linux / macOS
export PAOZIP_KEY="a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6"

# Windows
set PAOZIP_KEY=a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6
```

8.3 注意事項

・.paozip_key ファイルは 絶対に Git にコミットしないでください
・.gitignore に .paozip_key を追加してください
・暗号化と実行で必ず同じキーを使ってください — キーが違うと復号できません
・キーを紛失すると、暗号化されたファイルは復号できません — 大切に保管してください
・全環境（開発・ステージング・本番）で同じキーを使用してください

9. 各種環境での利用

9.1 Express

Express アプリケーションで使う場合は、エントリポイントの先頭でフックをインストールします：

```
// app.js —
const importer = require('paozip/lib/importer');
importer.install();

const express = require('express');
const routes = require('./routes');    // routes.jse
const auth = require('./lib/auth');    // lib/auth.jse

const app = express();
app.use('/', routes);
app.listen(3000, () => console.log('Server running on port 3000'));
```

```
#
paozip encrypt routes.js
paozip encrypt lib/auth.js
rm routes.js lib/auth.js

# Express
node app.js
```

9.2 npm パッケージとして配布

暗号化したモジュールを npm パッケージとして配布することもできます：

```
{
  "name": "my-secret-package",
  "main": "lib/index.js",
  "files": ["lib/*.jse", "lib/index.js"],
  "dependencies": {
    "paozip": "^2.0.0"
  }
}
```

```
// lib/index.js —
const importer = require('paozip/lib/importer');
importer.install();

module.exports = require('./core');  // core.jse
```

9.3 Docker

```
FROM node:20-slim

WORKDIR /app

# paozip
COPY paozip-nodejs/ /tmp/paozip-nodejs/
RUN cd /tmp/paozip-nodejs && npm install && npm link \
    && rm -rf /tmp/paozip-nodejs/

#
COPY . .
RUN npm install

#
RUN paozip encrypt lib/secret.js && rm lib/secret.js

CMD ["node", "app.js"]
```

ビルド時に暗号化し .js を削除すれば、配布イメージにソースは含まれません。

10. トラブルシューティング

症状	対処法
Encryption key not found	.paozip_key ファイルを作成するか、PAOZIP_KEY 環境変数を設定
Data is not encrypted (missing magic header)	暗号化されていないファイルを復号しようとしている。paozip check で確認
Cannot find module './secret'	importer.install() が呼ばれていないか、.jse ファイルが存在しない
.js と .jse の両方がある	.js が優先されます。暗号化後は .js を削除してください
復号後の内容が文字化けする	暗号化時と同じキーを使用しているか確認
paozip コマンドが見つからない	npm install -g paozip でグローバルインストール。または npx paozip を使用

11. 使用許諾

paozip for Node.js の使用について、paozip for Node.js の使用者（以下「利用者様」と称します）と有限会社パオ・アット・オフィス（以下「弊社」と称します）は、以下の各項目についての内容に同意するものとします。

1. 使用許諾書

この使用許諾書は、利用者様が paozip for Node.js を使用する場合に同意しなければならない契約書です。

2. 使用許諾書の同意

利用者様が paozip for Node.js を使用する時点で、本使用許諾書に同意されたものとします。同意されない場合は、paozip for Node.js を使用する事はできません。

3. ライセンス（使用権）の購入

利用者様が paozip for Node.js の製品版を使用して開発を行う場合には、1台の開発用コンピュータで paozip for Node.js を使用するにあたり、1ライセンスを購入する必要があります。

お客様環境等、開発コンピュータでないマシンで paozip for Node.js を使用する場合ライセンスは必要ありません。ランタイムライセンスフリーでございます。

4. 著作権

paozip for Node.js の著作権は、いかなる場合においても弊社に帰属いたします。

5. 免責

paozip for Node.js の使用によって、直接的、あるいは、間接的に生じた、いかなる損害に対しても、弊社は補償賠償の責任を負わないものとします。

6. 禁止事項

paozip for Node.js 及びその複製物を第三者に譲渡・貸与する事は出来ません。paozip for Node.js を開発ツールとして再販/再配布することを禁止します。なお、暗号化されたファイルを配布することは問題ございません。

7. 保証の範囲

弊社は paozip for Node.js の仕様を予告無しに変更することがあります。その場合の利用者様に対する情報提供は、弊社 WEB サイトにて行う事とします。

8. 適用期間

本使用許諾条件は利用者様が paozip for Node.js を使用した日より有効です。

12. 代金支払い方法

paozip for Node.js の製品版をご利用頂ける場合は、ライセンスを購入して頂く必要があります。

体験版について: すべての機能を制限なくお試しいただけます。体験版では [TRIAL] メッセージが表示されます。

必要なライセンス数の数え方

paozip for Node.js で開発を行うパソコンの台数

1ライセンス当たりの価格

11,000円（税込）

バグフィックス等のバージョンアップは原則として無償とさせて頂きます。

お支払方法

11,000円 × ライセンス数 を下記口座へ銀行振込、または、郵便振替による送金をして下さい。

銀行名	支店名（コード）	口座番号	名義
三菱UFJ銀行	新宿支店 (341)	普通 3831891	ユ）パオアットオフィス
PayPay銀行	すずめ支店 (002)	普通 6461359	ユ）パオアットオフィス

郵便口座番号	名義
00150-0-576845	有限会社 パオ・アット・オフィス

※ 振込手数料は利用者様負担でお願い致します。

お支払いの通知と製品の送付

1. 振り込みが完了した時点で、必ず弊社 WEB サイトの入金連絡フォームから入金のご連絡をお願いいたします。
2. 弊社では上記連絡を受けて入金確認を行い、paozip for Node.js の製品版を利用者様へ電子メールにてお送りさせて頂きます。

見積書/納品書/請求書/領収証の発行

見積書/納品書/請求書/領収証の発行は可能でございます。製品サイトでの手続きにより発行いたします。

お問い合わせ

製品に関するお問い合わせは、下記までお願いいたします。

有限会社 パオ・アット・オフィス

・製品サイト: https://www.pao.ac/paozip/

・購入ページ: https://www.pao.ac/paozip/buy.html

・メール: info@pao.ac

paozip for Node.js — あなたの JavaScript コードを、シンプルに、確実に、守ります。