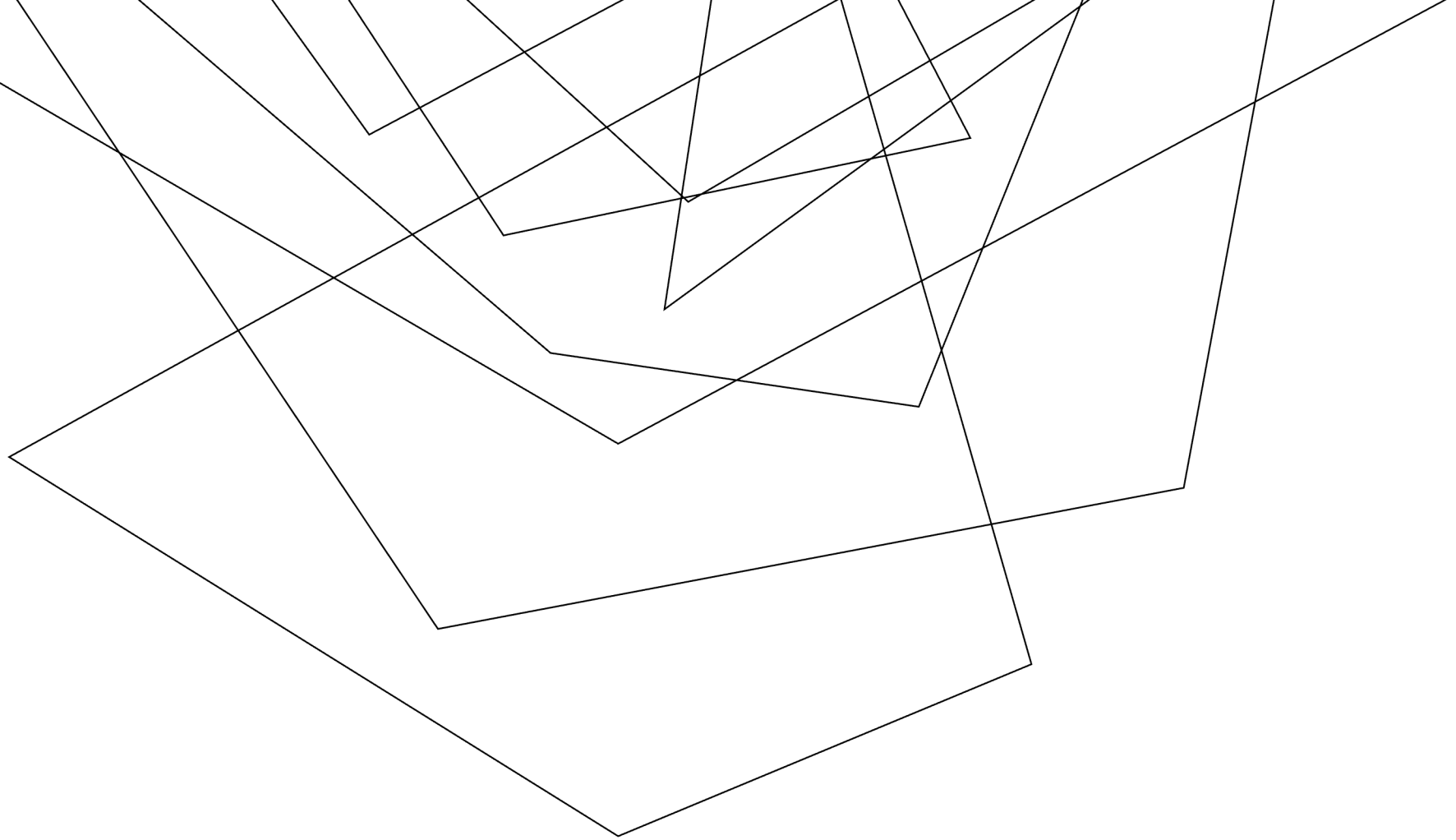
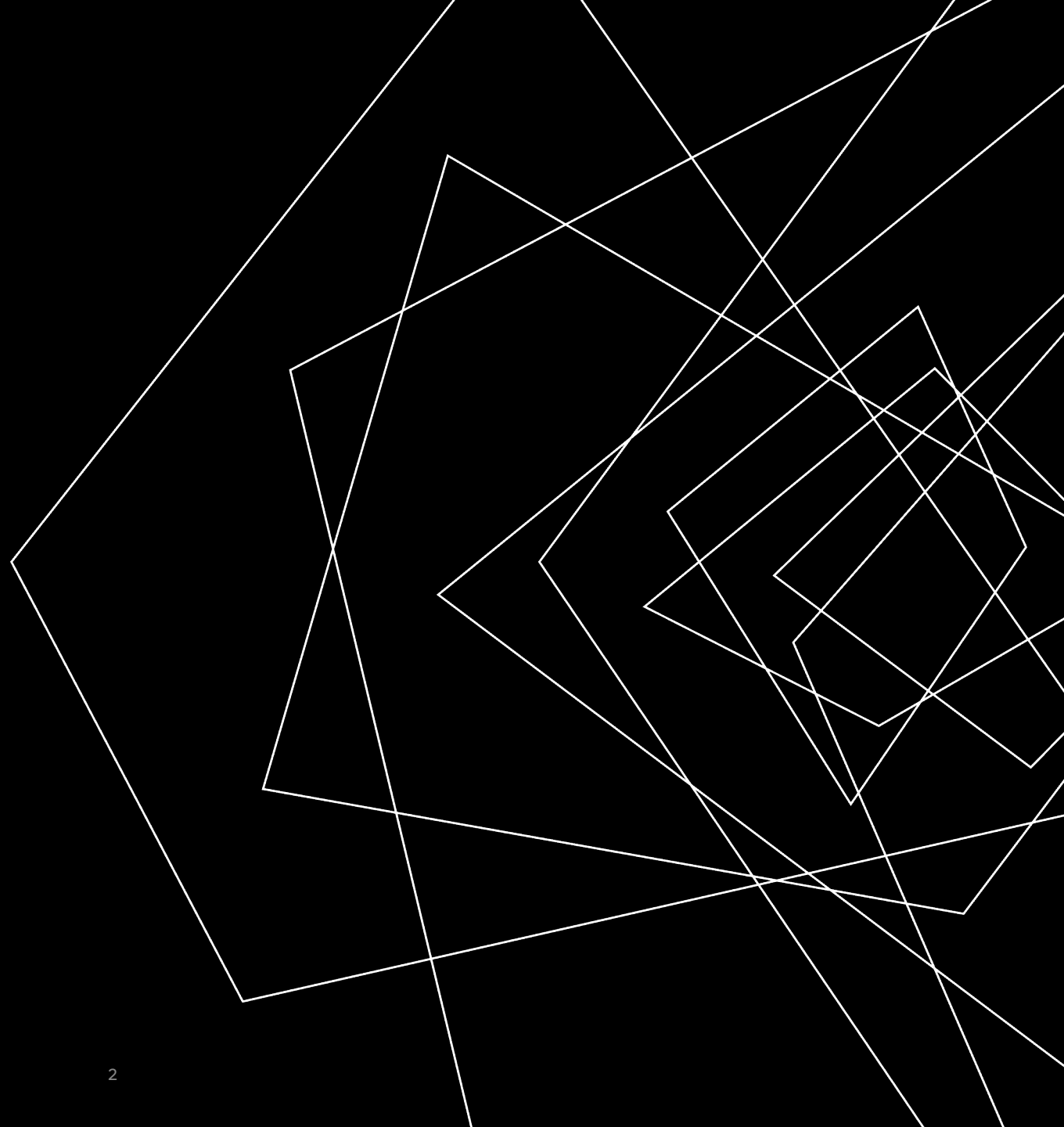


JSON WEB TOKEN

Muhammad Zulfi Izzulhaq

APA ITU JSON WEB TOKEN?

Json Web Token atau JWT adalah standar (RFC 7519) yang digunakan untuk pertukaran informasi secara aman dalam bentuk JSON yang sudah ditanda tangani secara digital.



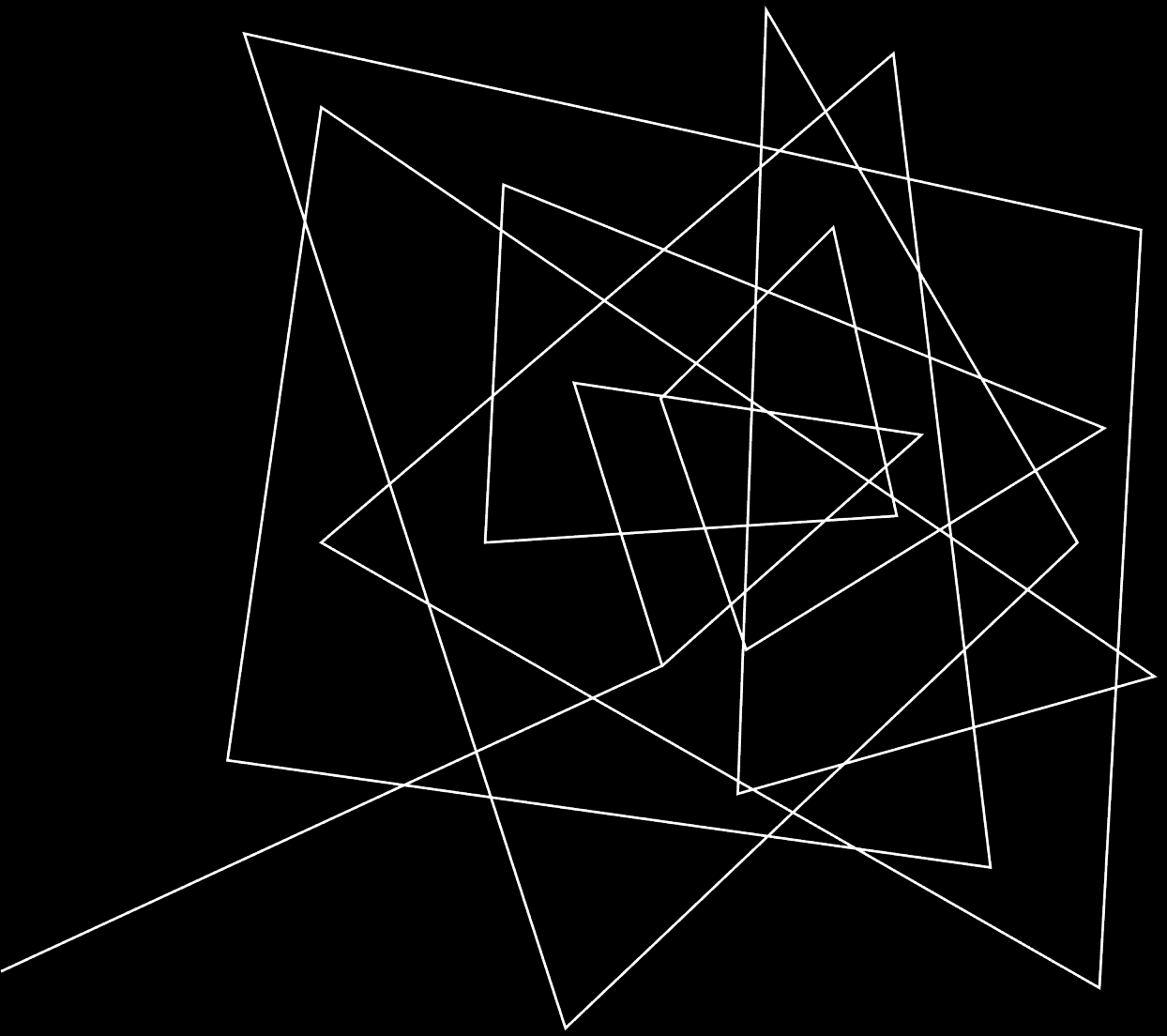
KAPAN HARUS MENGGUNAKAN JWT?

- **Authorization**

JWT sering digunakan untuk proses otorisasi aplikasi. Setelah pengguna berhasil login maka akan mendapat token yang akan dibawa setiap melakukan *request* ke aplikasi.

- **Information Exchange**

Selain digunakan untuk proses otorisasi sebuah sistem, JWT juga bisa digunakan untuk pertukaran infomasi karena memiliki *signature* yang bisa digunakan untuk memvalidasi keaslian data yang dikirim.



STRUKTUR JWT

header.payload.signature

Header
Header berisi jenis token yaitu JWT itu sendiri, dan jenis algoritma yang digunakan seperti HMAC SHA256 atau RSA

Payload
Berisi claims atau data yang akan dikirimkan seperti id, nama, ataupun email. Adapun data yang tidak boleh dikirimkan adalah data yang sifatnya rahasia atau sensitif seperti password atau *secret key*.
Terdapat 3 jenis JWT claims yaitu Registered Claims, Public Claims, dan Private Claims.

- Registered Claims: claims ini sifatnya tidak wajib namun direkomendasikan seperti iss (issuer), exp (expiration time), sub (subject), aud (audience).
- Public Claims: berisi informasi umum seperti nama, email, nomor telepon, dll.
- Private Claims: berisi informasi spesifik dari aplikasi misalnya ID Karyawan, ID Group, dll.

Signature
Signature adalah hasil hash dari header dan payload yang sudah diencode dengan base64. Signature digunakan untuk memverifikasi bahwa header maupun payload yang ada dalam token tidak berubah.

HEADER

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

```
16
17    type Header struct {
18        Algorithm string `json:"alg"`
19        Type      string `json:"typ"`
20    }
21
```

```
33
34  func createJWT(userId int64, email string) (string, error) {
35
36      // create the header
37      header := Header{
38          Algorithm: "HS256",
39          Type:      "JWT",
40      }
41
42      // encode the header as a JSON string
43      headerBytes, err := json.Marshal(header)
44      if err != nil {
45          return "", err
46      }
47
48      // base64 encode the header
49      base64UrlHeader := base64UrlEncode(headerBytes)
50
```

PAYLOAD

```
21
22  type RegisteredClaims struct {
23      Issuer      string `json:"iss"`
24      IssuedAt    int64  `json:"iat"`
25      ExpiresAt   int64  `json:"exp"`
26  }
27
28  type Claims struct {
29      UserID int64  `json:"user_id"`
30      Email  string `json:"email"`
31      RegisteredClaims
32  }
33
```

```
50
51      // create claims for jwt
52      claims := Claims{
53          UserID: userId,
54          Email:  email,
55          RegisteredClaims: RegisteredClaims{
56              Issuer:    "zlfzx",
57              IssuedAt:  time.Now().Unix(),
58              ExpiresAt: time.Now().Add(time.Hour * 24).Unix(),
59          },
60      }
61
62      // encode the claims as a JSON string
63      claimsBytes, err := json.Marshal(claims)
64      if err != nil {
65          return "", err
66      }
67
68      // base64 encode the claims
69      base64UrlClaims := base64UrlEncode(claimsBytes)
70
```

eyJ1c2VyX2lkIjoxMCwiZW1haWwiOiJ1c2VyQGVtYWlsLmNvbSIsImlzcyI6InpsZnp4IiwiaWF0IjoxNjczMzYxMjUyLCJleHAiOjE2NzM0NDc2NTJ9

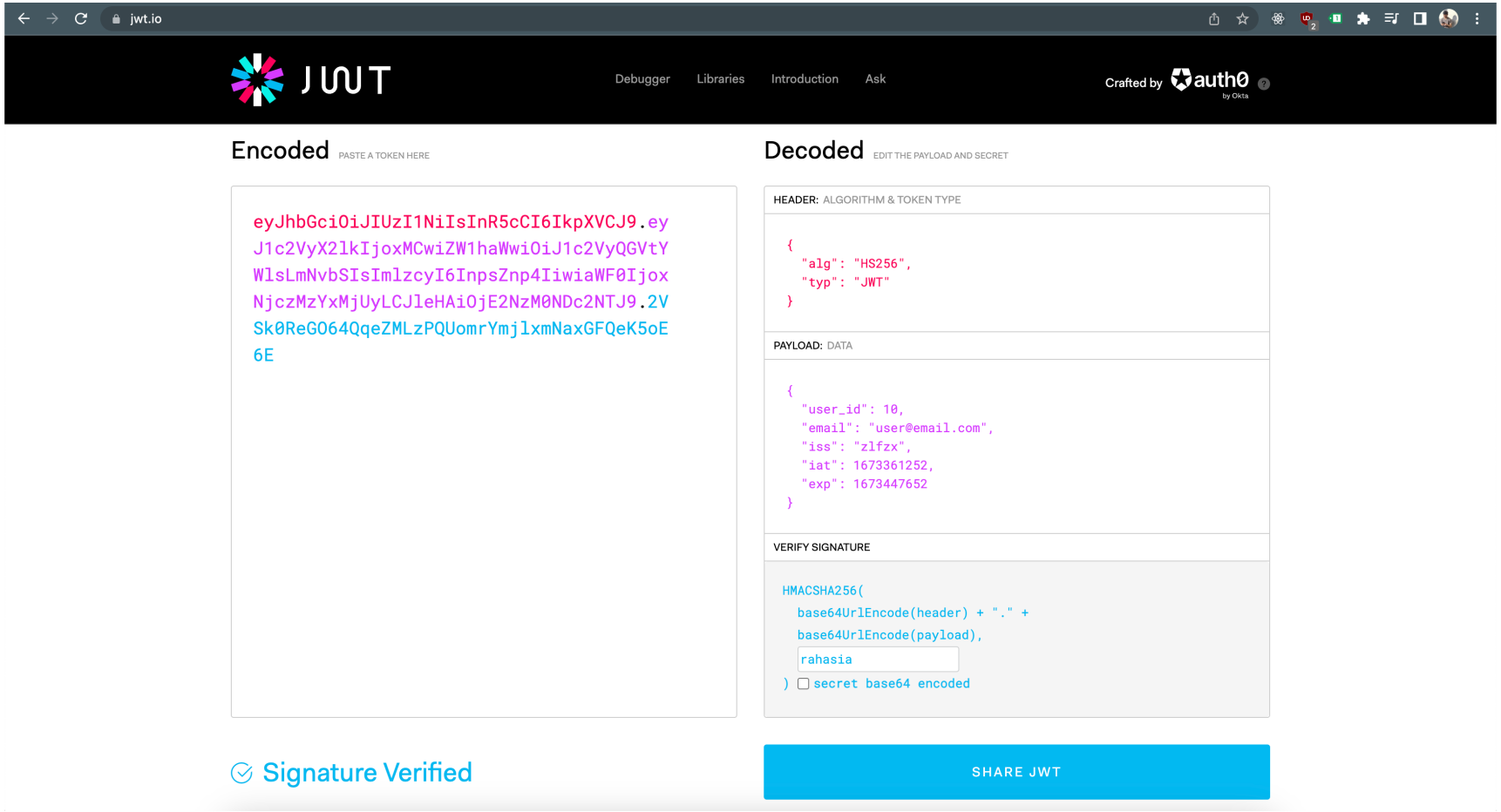
SIGNATURE

```
70
71      // create signature
72      signature := hmac.New(sha256.New, []byte(secret))
73      signature.Write([]byte(base64UrlHeader + "." + base64UrlClaims))
74
75      // base64 encode the signature
76      base64UrlSignature := base64UrlEncode(signature.Sum(nil))
77
78      // construct the JWT
79      token := base64UrlHeader + "." + base64UrlClaims + "." + base64UrlSignature
80
81      return token, nil
82  }
83
```

```
14
15  const secret = "rahasia"
16
```

Setelah Header, Payload, dan Signature terbentuk selanjutnya digabungkan dengan tanda “titik” sebagai pemisahnya.

2VSk0ReGO64QqeZMLzPQUomrYmjlxmNaxGFQeK5oE6E



eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoxMCwiZW1haWwiOiJ1c2VyQGVtYWlsLmNvbSIsImlzcyI6InpsZnp4IiwiaWF0IjoxNjczMzYxMjUyLCJleHAiOjE2NzM0NDc2NTJ9.2VSk0ReGO64QqeZMLzPQUomrYmjlxmNaxGFQeK5oE6E

VERIFIKASI JWT

Karena JWT diencode menggunakan bas64, maka untuk melihat isi datanya bisa didecode dengan base64 decode.

```
92
93  func verifyJWT(token string) (Claims, error) {
94      var header Header
95      var claims Claims
96
97      jwt := strings.Split(token, ".")
98      if len(jwt) != 3 {
99          return claims, errors.New("invalid JWT format!")
100     }
101
102     headerEncode, claimsEncode, signatureEncode := jwt[0], jwt[1], jwt[2]
103
104     headerBytes, err := base64UrlDecode(headerEncode)
105     if err != nil {
106         return claims, err
107     }
108     if err = json.Unmarshal(headerBytes, &header); err != nil {
109         return claims, err
110     }
111
112     claimsBytes, err := base64UrlDecode(jwt[1])
113     if err != nil {
114         return claims, err
115     }
116     if err = json.Unmarshal(claimsBytes, &claims); err != nil {
117         return claims, err
118     }
119
```

```
119
120     // create signature
121     signature := hmac.New(sha256.New, []byte(secret))
122     signature.Write([]byte(headerEncode + "." + claimsEncode))
123
124     expectedSignature := base64UrlEncode(signature.Sum(nil))
125
126     if signatureEncode != expectedSignature {
127         return claims, errors.New("invalid signature!")
128     }
129
130     if claims.ExpiresAt < time.Now().Unix() {
131         return claims, errors.New("token expired!")
132     }
133
134     return claims, nil
135 }
136
```

Buat signature dari header dan payload lalu cek apakah hasilnya sama dengan signature dari token JWT tersebut.

Jika payload sudah pernah dirubah maka hasil signature yang baru akan berbeda yang berarti token tidak valid.

Setelah pengecekan signature, selanjutnya bisa mengecek bagian lain seperti expiration time untuk mengecek apakah masa aktif token masih berlaku atau tidak.

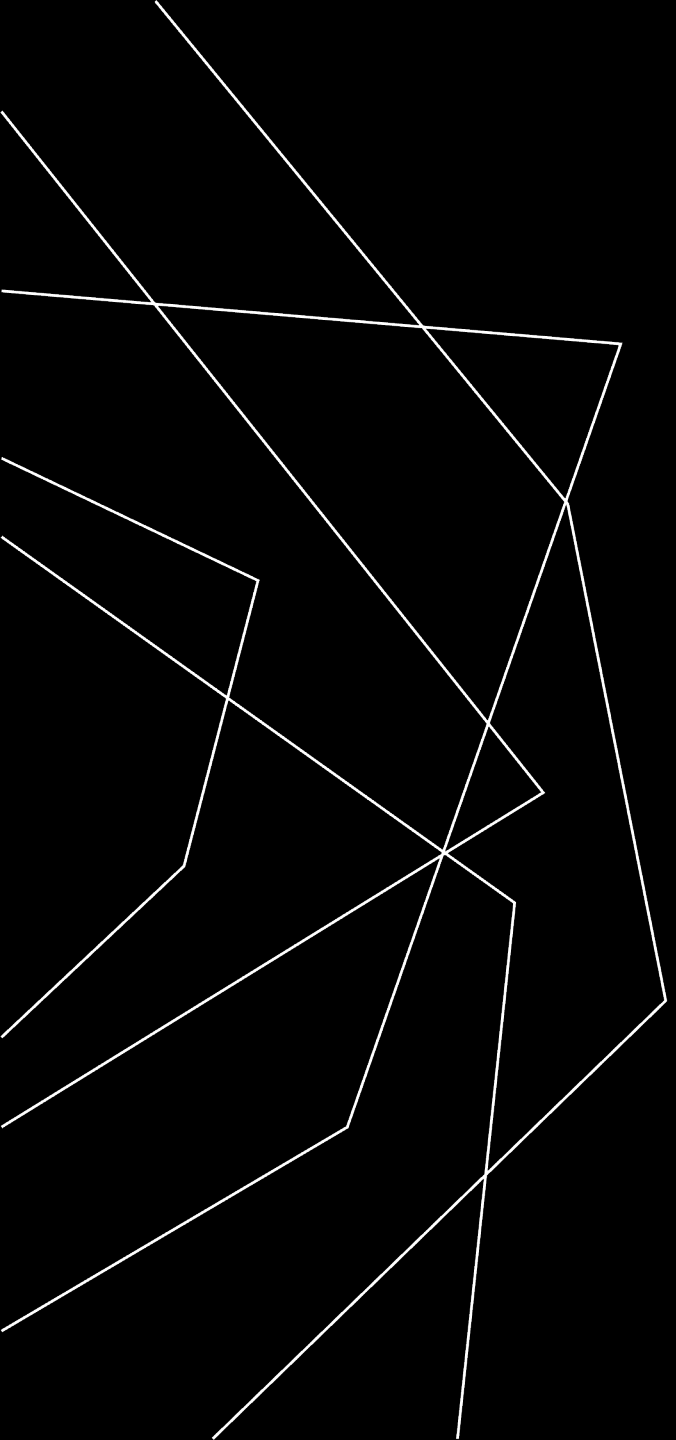
LIBRARIES

GO
https://github.com/golang-jwt/jwt

Python
https://github.com/jpadilla/pyjwt

PHP
https://github.com/firebase/php-jwt

NodeJS
https://github.com/auth0/node-jsonwebtoken



THANK YOU

👉 https://github.com/zlfzx/what-is-jwt