# $SAS\text{-}M_{\mathsf{izoguchi}}$

#### Simple And Secure Mutual Authentication Protocol

#### MIZOGUCHI Koki<sup>1</sup>

Kochi University of Technology

November 16, 2022



#### **KOCHI UNIVERSITY OF TECHNOLOGY**

K.MIZOGUCHI (KUT) SAS-M November 16, 2022

## 特徴

SAS-M(仮)は、SAS-Lを基盤として、Client と Server が相互認証する機能を追加した認証プロトコル.

# 相互認証の必要性

SAS-Lでは、ServerからClientの認証は可能だが、Serverから登録されているClientはServerがが正当なものとして通信が行われる。つまり、ClientはServerを真には認証していない。Serverを真に認証するとこで、重要な情報を送るClientはServerの正当性を確かめることに十分な意味を見出せる。

# 認証手順

## Server 生成データ

$$A_i = E_1(\text{SID} \mid S \oplus N_i)$$

$$A_{i+1} = E_1(\text{CID} \mid S \oplus N_{i+1})$$

$$A_{i+2} = E_1(\text{CID} \mid S \oplus N_{i+2})$$

### 初回認証情報

$$A_i \oplus A_{i+1}$$
 1. Secure to Client  $eta_i = E_1ig(A_i \oplus A_{i+1}ig)$  2 to Client  $eta_{i+1} = ig(A_i \oplus A_{i+2}ig)$ 

送信用データ

<sup>1</sup>SID:サーバ固有 ID

<sup>2</sup>CID:クライアント固有 ID

## Server 生成データ

$$A_i = E_1(\text{SID} \mid S \oplus N_i)$$

$$A_{i+1} = E_1(\text{CID} \mid S \oplus N_{i+1})$$

$$A_{i+2} = E_1(\text{CID} \mid S \oplus N_{i+2})$$

#### 初回認証情報

$$A_i \oplus A_{i+1}$$

1.Secure to Client

$$\beta_i = E_1 \Big( A_i \oplus A_{i+1} \Big)$$

2

$$\beta_{i+1} = \left( A_i \oplus A_{i+2} \right)$$

送信用データ

......

<sup>1</sup>SID:サーバ固有 ID

<sup>2</sup>CID:クライアント固有 ID

## Server 生成データ

$$A_i = E_1(SID \mid S \oplus N_i)$$

$$A_{i+1} = E_1(\text{CID} \mid S \oplus N_{i+1})$$

$$A_{i+2} = E_1(\text{CID} \mid S \oplus N_{i+2})$$

#### 初回認証情報

$$A_i \oplus A_{i+1}$$
 1. Secure to Client

1.Secure

4/8

$$\beta_i = E_1 \Big( A_i \oplus A_{i+1} \Big)$$

$$\beta_{i+1} = \left( A_i \oplus A_{i+2} \right)$$

送信用データ

<sup>2</sup>CTD:クライアント固有 ID

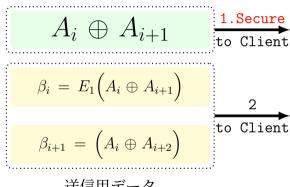
### Server 生成データ

$$A_i = E_1(\text{SID} \mid S \oplus N_i)$$

$$A_{i+1} = E_1(\text{CID} \mid S \oplus N_{i+1})$$

$$A_{i+2} = E_1(\text{CID} \mid S \oplus N_{i+2})$$

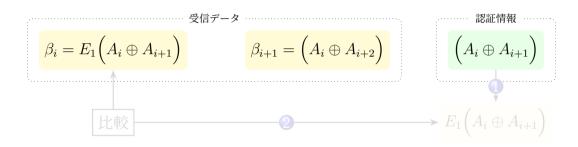
#### 初回認証情報



送信用データ

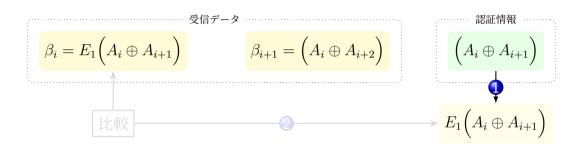
<sup>1</sup>SID:サーバ固有 ID

<sup>2</sup>CTD:クライアント固有 ID



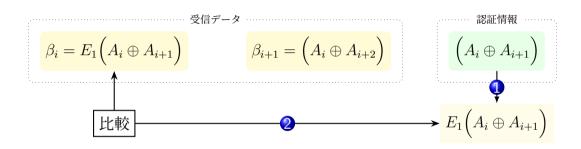
- ◎ 認証情報に一方向ハッシュ関数を施す.
- ② 比較 で Client が Server を検証する.

• 不一致の場合、認証不成立. (Server が不正である可能性あり)

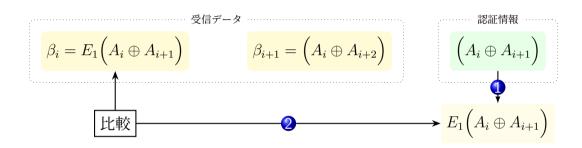


- 認証情報に一方向ハッシュ関数を施す.
- ② 比較 で Client が Server を検証する.

• 不一致の場合、認証不成立. (Server が不正である可能性あり)



- 認証情報に一方向ハッシュ関数を施す.
- ② 比較 で Client が Server を検証する.
  - 不一致の場合、認証不成立. (Server が不正である可能性あり)



- 認証情報に一方向ハッシュ関数を施す.
- ② 比較 で Client が Server を検証する.
  - 不一致の場合、認証不成立. (Server が不正である可能性あり)

受信データ

$$\beta_i = E_1 \Big( A_i \oplus A_{i+1} \Big)$$

$$\beta_{i+1} = \left( A_i \oplus A_{i+2} \right)$$

認証情報 ………

$$(A_i \oplus A_{i+1})$$

$$\beta_{i+1} \oplus \left( A_i \oplus A_{i+1} \right) = \left( A_i \oplus A_{i+2} \right) \oplus \left( A_i \oplus A_{i+1} \right)$$
$$= A_{i+1} \oplus A_{i+2}$$

- ③ 次回認証情報,  $A_{i+1} \oplus A_{i+2}$  を Client に保存.
- $\gamma_i = E_2(A_{i+1} \oplus A_{i+2})$ を生成.
- ⑤  $\gamma_i$  を Server へ送信.

$$\beta_{i} = E_{1} \left( A_{i} \oplus A_{i+1} \right)$$

$$\beta_{i+1} = \left( A_{i} \oplus A_{i+2} \right)$$

$$= \left( A_{i} \oplus A_{i+1} \right)$$

$$= \left( A_{i} \oplus A_{i+1} \right)$$

$$= \left( A_{i} \oplus A_{i+2} \right) \oplus \left( A_{i} \oplus A_{i+1} \right)$$

$$= \left( A_{i} \oplus A_{i+2} \right)$$

認証情報 ………

⑤ 次回認証情報,  $A_{i+1} \oplus A_{i+2}$  を Client に保存.

受信データ

- $\gamma_i = E_2(A_{i+1} \oplus A_{i+2})$ を生成.
- ⑤  $\gamma_i$  を Server へ送信.

$$\beta_{i} = E_{1} \left( A_{i} \oplus A_{i+1} \right) \qquad \beta_{i+1} = \left( A_{i} \oplus A_{i+2} \right) \qquad \left( A_{i} \oplus A_{i+1} \right)$$

$$\beta_{i+1} \oplus \left( A_{i} \oplus A_{i+1} \right) = \left( A_{i} \oplus A_{i+2} \right) \oplus \left( A_{i} \oplus A_{i+1} \right)$$

 $=A_{i+1}\oplus A_{i+2}$ 

認証情報 ………

③ 次回認証情報,  $A_{i+1} \oplus A_{i+2}$  を Client に保存.

受信データ

- $\gamma_i = E_2(A_{i+1} \oplus A_{i+2})$  を生成.
- ⑤  $\gamma_i$ を Server へ送信.

$$\beta_{i} = E_{1} \left( A_{i} \oplus A_{i+1} \right) \qquad \beta_{i+1} = \left( A_{i} \oplus A_{i+2} \right) \qquad \left( A_{i} \oplus A_{i+1} \right)$$

$$\beta_{i+1} \oplus \left( A_{i} \oplus A_{i+1} \right) = \left( A_{i} \oplus A_{i+2} \right) \oplus \left( A_{i} \oplus A_{i+1} \right)$$

$$(3)$$

 $=A_{i+1}\oplus A_{i+2}$ 

認証情報 ………

⑤ 次回認証情報,  $A_{i+1} \oplus A_{i+2}$  を Client に保存.

受信データ

- $\circ$   $\gamma_i$  を Server へ送信.

受信データ 生成済みデータ 
$$\gamma_i = E_2\Big(A_{i+1} \oplus A_{i+2}\Big) \hspace{1cm} A_i \hspace{1cm} A_{i+1} \hspace{1cm} A_{i+2}$$

- $\odot$  生成済みデータから, $E_2\left(A_{i+1} \oplus A_{i+2}\right)$  を生成する.
- ② 受信データ $\gamma_i$ と比較する.
  - 不一致の場合,認証不成立.(Client が不正である可能性あり.)
  - 一致した場合,認証成立。
- ◎ γ<sub>i</sub>を利用した共通鍵暗号方式で通信を行う.

受信データ 生成済みデータ  $\gamma_i = E_2\Big(A_{i+1} \oplus A_{i+2}\Big) \hspace{1cm} A_i \hspace{1cm} A_{i+1} \hspace{1cm} A_{i+2}$ 

- ullet 生成済みデータから, $E_2\Big(A_{i+1}\oplus A_{i+2}\Big)$  を生成する.
- 受信データ γ<sub>i</sub> と比較する.
   不一致の場合,認証不成立. (Client が不正である可能性あり.)
   一致した場合,認証成立.
- ◎ γ<sub>i</sub>を利用した共通鍵暗号方式で通信を行う.

受信データ 
$$\gamma_i = E_2\Big(A_{i+1} \oplus A_{i+2}\Big)$$

------ 生成済みデータ

 $A_{i+1}$ 

 $A_{i+2}$ 

- $\bullet$  生成済みデータから, $E_2\Big(A_{i+1}\oplus A_{i+2}\Big)$  を生成する.
- ② 受信データ $\gamma_i$ と比較する.
  - 不一致の場合,認証不成立. (Client が不正である可能性あり.)
  - 一致した場合, 認証成立.
- ◎ γ<sub>i</sub>を利用した共通鍵暗号方式で通信を行う.

受信データ  $\gamma_i = E_2\Big(A_{i+1} \oplus A_{i+2}\Big)$ 

 $A_i$ 

 $A_{i+1}$ 

----- 生成済みデータ

 $A_{i+2}$ 

- $\bullet$  生成済みデータから, $E_2\Big(A_{i+1}\oplus A_{i+2}\Big)$  を生成する.
- 受信データ γ<sub>i</sub> と比較する.
  - 不一致の場合, 認証不成立. (Client が不正である可能性あり.)
  - 一致した場合, 認証成立.
- ◎ γ<sub>i</sub>を利用した共通鍵暗号方式で通信を行う.

受信データ 生成済みデータ  $\gamma_i = E_2\Big(A_{i+1} \oplus A_{i+2}\Big)$   $A_i$   $A_{i+1}$   $A_{i+2}$ 

- ullet 生成済みデータから, $E_2\Big(A_{i+1}\oplus A_{i+2}\Big)$  を生成する.
- 受信データ γ<sub>i</sub> と比較する.
  - 不一致の場合, 認証不成立. (Client が不正である可能性あり.)
  - 一致した場合, 認証成立.

# 軽量度

Server と Client の一方向ハッシュ関数の利用回数・排他的論理和の排他的論理和の演算回数は以下.

演算	Client	Server
一方向ハッシュ関数	3	6
排他的論理和	1	6

結果のように、Clientでは一方向性ハッシュ関数の適用が3回である. これは、SAS-Lの**0回**と比べて軽量とは言えない. Clientもある程度の処理能力は必要であるう.

K.MIZOGUCHI (KUT) SAS-M November 16, 2022 8/8