Vol.2018-DPS-174 No.22
Vol.2018-CSEC-80 No.22
2018/3/6

ハッシュ連鎖アグリゲーションを用いた共通鍵作成法における結託攻撃への対策

栗原 勇太[1,a)]　双紙 正和[1,b)]

概要：IoT (Internet of things) の普及に伴い，計算能力やメモリ，バッテリ等に制約のあるデバイスをネットワーク接続する機会が増加しており，ハッシュ関数を利用した軽量かつ効率の良いプロトコルの需要が高まっている．そこで，新しいハッシュ連鎖構成法 *hash chain aggregation* (HCA)，及び，それを用いた，二者間での共通鍵生成法が提案された．*hash chain aggregation* (HCA) は，ハッシュ関数のみを暗号プリミティブとして用いており効率が良いが，結託攻撃への対策が急務であった．本論文では，*hash chain aggregation* (HCA) における結託攻撃への対策法を提案する．

Countermeasures against Collusion Attacks on Secret Key Generation using Hash Chain Aggregations

Yuta Kurihara[1,a)]　Masakazu Soshi[1,b)]

Abstract: 'IoT' (Internet of Things) environments interconnect many IoT devices, many of which often suffer from limited computational resources. Thus in order to provide lightweight and efficient authentication protocols in such environments, we have proposed a novel hash chain construction, *hash chain aggregation* (HCA). In this paper we propose the countermeasure against collusion attacks on secret key generation using HCA.

1. 概要

近年，Internet of Things (IoT) [1] と呼ばれるネットワーク環境が注目されている．IoT 環境では，計算能力やメモリ，バッテリ等に制約のあるデバイスを相互接続する機会が増加しており，ハッシュ関数などを用いた軽量かつ効率の良いプロトコルの需要が高まっている．

ここで，ハッシュ関数とは，一方向性と衝突困難性をもつ関数 (暗号プリミティブ) [2] である．一方で，ハッシュ関数は，量子計算機による攻撃にも耐えうると信じられており [3]，近年再び脚光を浴びている．現在, 量子コンピュータが効率的に素因数分解問題や離散対数問題を解いてしまうことは良く知られている [4]，多くの現代的暗号システムはそれらの困難性に基づいており，その対策が急務である．

このようなハッシュ関数の応用で重要なものに，ハッシュ連鎖がある．ハッシュ連鎖とは，入力にハッシュ関数を繰り返し適用して得られるハッシュ値の列であり，様々な暗号プロトコルの要素技術として用いられている [5], [6], [7], [8]．しかしながら，それらのプロトコルにおけるハッシュ連鎖の使用は，単純なものにとどまっており，ハッシュ連鎖のポテンシャルを生かしきれていない．また，これまでハッシュ関数のみを用いた相互認証プロトコルは知られていなかった．

そこで，全く新しく，かつ柔軟なハッシュ連鎖構成法 "ハッシュ連鎖アグリゲーション" が提案され，ハッシュ連鎖アグリゲーションを用いた，二者間での共通鍵生成法も提案されている [13]．ハッシュ連鎖アグリゲーションを用いた認証は，以下のような重要な利点を持つ:

(1) ハッシュ関数のみを暗号プリミティブとして用いており，効率がよい．
(2) 全く新しいハッシュ連鎖構成法 (ハッシュ連鎖アグリ

[1] 広島市立大学
Hiroshima City University
[a)] kurihara@sos.info.hiroshima-cu.ac.jp
[b)] soshi@hiroshima-cu.ac.jp

ゲーション) に基づいている．

(3) 相手の ID のみを知っていればよく，鍵生成に通信を必要としない，ID-based 認証プロトコルである．

残念ながら，この手法のセキュリティについては，2 人以上のユーザの結託によって，特定のユーザのペアの共通鍵を作成可能であることが判明している．

そこで，本論文では，この結託攻撃に対する対策法を提案する．

本論文は以下のように構成される．

2 節では，ハッシュ連鎖アグリゲーション，及びそれを用いた共通鍵作成法について説明し，3 節では，結託攻撃への対策法について議論する．最後に，4 節にて結論を述べる．

2. ハッシュ連鎖アグリゲーション

本節では，ハッシュ連鎖アグリゲーション，及びそれを用いた共通鍵作成法 [13] について説明する．

まず，ハッシュ連鎖を定義する．ハッシュ連鎖 (hash chain) とは，種 s にハッシュ関数 h を繰り返し適用して得られるハッシュ値の列である．ここで，$h^1(s) := h(s)$，$i \geq 2$ について，$h^i(s) := h(h^{i-1}(s))$ と定義する．このとき，長さ n のハッシュ連鎖は，集合 $\{1, 2, ..., n\}$ の一つの置換 $(i_1, i_2, ..., i_n)$ と，種 s を用いて，

$$(v_1, v_2, ..., v_n) \quad \text{について } v_j = h^{i_j}(s), 1 \leq j \leq n \quad (1)$$

と定義される．ここで，n 個のハッシュ値は，$h(s), h^2(s), \ldots, h^n(s)$ の順で計算されることに注意せよ．本論文では，それらのハッシュ値を，任意の順で並べ替えたものを，ハッシュ連鎖と呼ぶ．

このプロトコルにおけるプレイヤーは，Key Generation Center (KGC) と，N 人のユーザである．KGC は信頼できるものとし，KGC とそれぞれのユーザの間には，安全なチャンネルがあるとする

2.1 基本的なハッシュ連鎖

以降では，記述を簡単にするため，ある正整数 m について，$N = 2^m$ と仮定する．また，ある系列 $Q = (q_1, q_2, \ldots, q_j)$ について，i 番目の要素 q_i $(1 \leq i \leq j)$ を $Q[i]$ と書くことにする．

ハッシュ連鎖構成法を定義するために，まずタイプ I からタイプ IV までのハッシュ連鎖を定義する．以降では，ある正整数 b について，ハッシュ連鎖の長さを $\ell = 2^b$ $(1 \leq b \leq m)$ と仮定し，それぞれのハッシュ連鎖を式 (1) のような列として考える．また以下の定義では，$1 \leq i \leq \ell$ とし，s をハッシュ連鎖の seed とする．また，以降では，ハッシュ関数を h とする．

- タイプ I ハッシュ連鎖

$$C_\ell^I(s) := (v_1, v_2, \ldots, v_\ell) \quad \text{where } v_i = h^i(s)$$

- タイプ II ハッシュ連鎖

$$C_\ell^{II}(s) := (v_1, v_2, \ldots, v_\ell) \quad \text{where } v_i = h^{\ell-i+1}(s)$$

- タイプ III ハッシュ連鎖

$$C_\ell^{III}(s) := (v_1, v_2, \ldots, v_\ell)$$
$$\text{where } v_i = h^{((i-\ell/2-1) \bmod \ell)+1}(s)$$

- タイプ IV ハッシュ連鎖

$$C_\ell^{IV}(s) := (v_1, v_2, \ldots, v_\ell)$$
$$\text{where } v_i = h^{((\ell/2-i) \bmod \ell)+1}(s)$$

2.2 ハッシュ連鎖リスト

2.1 節の定義を用いれば，ハッシュ連鎖リスト (HCL) $\mathcal{L}(\tau, k, s_1, ..., s_k)$ を定義することが出来る．

$$\mathcal{L}(\tau, k, s_1, ..., s_k) := (C^\tau_{N/k}(s_1), C^\tau_{N/k}(s_2), ..., C^\tau_{N/k}(s_k)) \quad (2)$$

ここで，ある正整数 u (ただし $0 \leq u < m$) が存在して，$k = 2^u$ である．また，$\tau \in \{I, II, III, IV\}$ であり，$s_1, ..., s_k$ を，種の列とする．以降では，簡便さのため，$\mathcal{L}(\tau, k, s_1, ..., s_k)$ の $\tau, k, s_1, ..., s_k$ を省略して，単に $\mathcal{L}$ などと書くことがある．また，ハッシュ連鎖リスト $\mathcal{L}$ が与えられたとき，それに属するハッシュ連鎖の数を $k_\mathcal{L}$ と表すことがある．

2.3 ハッシュ連鎖アグリゲーション

ハッシュ連鎖リストを定義したことにより，r 個のハッシュ連鎖リストの集合として，ハッシュ連鎖アグリゲーション (HCA) $\mathcal{A}$ を定義することが出来る．

$$\mathcal{A} := \{\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_r\} \quad (3)$$

$\mathcal{L}_i$ $(1 \leq i \leq r)$ は，式 (2) によって定義される．

2.4 ユーザに割り当てられるハッシュ値

次に，HCA において，ノード i に割り当てられるハッシュ値の集合を考えよう．

まず，式 (3) における任意の $\mathcal{L}$ $(\in \mathcal{A})$ について考える．$\mathcal{L}$ においては，長さ $N/k_\mathcal{L}$ $(= 2^{m-u})$ のハッシュ連鎖が $k_\mathcal{L}$ 個あるので，$\mathcal{L}$ は N 個のハッシュ値を持つ．すなわち，任意のユーザ i $(1 \leq i \leq N)$ は，$\mathcal{L}$ のあるハッシュ連鎖から，一つのハッシュ値が割り当てられる．このハッシュ値は，以下のようにして求めることができる．

まず，

$$\frac{N}{k_\mathcal{L}}(d-1) + 1 \leq i \leq \frac{N}{k_\mathcal{L}} d \quad (1 \leq d \leq k_\mathcal{L}) \ . \quad (4)$$

式 (4) を満たす d が一意に定まることに注意せよ．そこ

で，このような d を，$\alpha(\mathcal{L}, i)$ と書く

すると，$\mathcal{L}$ において，ユーザ i に割り当てられるハッシュ値は，

$$(\mathcal{L}[\alpha(\mathcal{L}, i)])\left[i - \frac{N}{k_{\mathcal{L}}} \cdot (\alpha(\mathcal{L}, i) - 1)\right]$$

となる

以上より，HCA $\mathcal{A}$ においてユーザ i に割り当てられるハッシュ値の集合は，

$$V(\mathcal{A}, i) := \left\{(\mathcal{L}[\alpha(\mathcal{L}, i)])\left[i - \frac{N}{k_{\mathcal{L}}} \cdot (\alpha(\mathcal{L}, i) - 1)\right] \;\middle|\; \mathcal{L} \in \mathcal{A}\right\} . \tag{5}$$

と定義される関数 V によって表現できる．KGC は，ユーザ i に対し $V(\mathcal{A}, i)$ を安全なチャンネルを介して送る．ユーザ i は，送られてきた値を秘密に保持しておく．

また，$\mathcal{A}$ においてユーザ i を含むハッシュ連鎖の集合は，

$$W(\mathcal{A}, i) := \{\mathcal{L}[\alpha(\mathcal{L}, i)] \mid \mathcal{L} \in \mathcal{A}\} \tag{6}$$

と定義される関数 W によって表現できる．

2.5 共通鍵作成法のためのハッシュ連鎖アグリゲーション

共通鍵作成法のためのハッシュ連鎖アグリゲーション $\mathcal{HCA}$ を以下のように定義する．なお以下では，ハッシュ連鎖リスト $\mathcal{L}(\tau, k, s_1, ..., s_k)$ の $\tau, k, s_1, ..., s_k$ を省略する．$N = 2^m$ であること注意せよ．

$$\mathcal{HCA} := (\mathcal{L}_{(1)}, \mathcal{L}_{(2)}, \ldots, \mathcal{L}_{(2m-1)}, \mathcal{L}_{(2m)}) \tag{7}$$

ここで，$\mathcal{L}_{(1)} = (C_N^I(s_1))$, $\mathcal{L}_{(2)} = (C_N^{II}(s_2))$, $\mathcal{L}_{(3)} = (C_N^{III}(s_3))$, $\mathcal{L}_{(4)} = (C_N^{IV}(s_4))$ であり，$i > 4$ のときの $\mathcal{L}_{(i)}$ は以下のように定義される．以下では，$3 \le j \le m$ である．

1. $i = 2j - 1$ のとき:

$$\mathcal{L}_{(2j-1)} := (C^{III}_{2^{m-j+2}}(s_{2j-1,1}), C^{III}_{2^{m-j+2}}(s_{2j-1,2}), \cdots, C^{III}_{2^{m-j+2}}(s_{2j-1,2^{j-2}}))$$

2. $i = 2j$ のとき:

$$\mathcal{L}_{(2j)} := (C^{IV}_{2^{m-j+2}}(s_{2j,1}), C^{IV}_{2^{m-j+2}}(s_{2j,2}), \cdots, C^{IV}_{2^{m-j+2}}(s_{2j,2^{j-2}}))$$

ここで，整数 a, b, c について，$s_a, s_{b,c}$ はすべて異なるランダムな種を表す．

例として，$N = 8$ のときの $\mathcal{HCA} = (\mathcal{L}_{(1)}, \mathcal{L}_{(2)}, \ldots, \mathcal{L}_{(5)}, \mathcal{L}_{(6)})$ を，図 1 に示す．このとき，ユーザ 3 について考えると，$V(\mathcal{HCA}, 3) = \{h^3(s_1), h^6(s_2), h^7(s_3), h^2(s_4), h(s_{5,1}), h^4(s_{6,1})\}$ である．また，$W(\mathcal{HCA}, 3) = \{C_8^I(s_1), C_8^{II}(s_2), C_8^{III}(s_3), C_8^{IV}(s_4), C_4^{III}(s_{5,1}), C_4^{IV}(s_{6,1})\}$ である．

2.6 $\mathcal{HCA}$ による共通鍵生成

$\mathcal{HCA}$ を用いたユーザ i, j $(1 \le i < j \le N)$ による，相互認証のための共通鍵作成法は以下の通りである．

1. $i + 1 = j$ のとき: $F(\mathcal{L}_{(1)}[1])[j], (\mathcal{L}_{(2)}[1])[i])$ とすればよい．
2. それ以外のとき: ある正整数 $(2 \le) k_1 < k_2 < \cdots < k_u$ $(\le m)$ を用いて，

$$\begin{aligned} &W(\mathcal{HCA}, i) \cap W(\mathcal{HCA}, j) = \\ &\{C_N^I(s_1), C_N^{II}(s_2), \\ &\mathcal{L}_{(2k_1-1)}[\alpha(\mathcal{L}_{(2k_1-1)}, i)], \mathcal{L}_{(2k_1)}[\alpha(\mathcal{L}_{(2k_1)}, i)], \\ &\vdots \\ &\mathcal{L}_{(2k_u-1)}[\alpha(\mathcal{L}_{(2k_u-1)}, i)], \mathcal{L}_{(2k_u)}[\alpha(\mathcal{L}_{(2k_u)}, i)]\} \end{aligned}$$

と書くことが出来る．ここで，$\alpha(\mathcal{L}_{(2k_1-1)}, i) = \alpha(\mathcal{L}_{(2k_1-1)}, j) = \alpha(\mathcal{L}_{(2k_1)}, i) = \alpha(\mathcal{L}_{(2k_1)}, j)$, $\cdots$, $\alpha(\mathcal{L}_{(2k_u-1)}, i) = \alpha(\mathcal{L}_{(2k_u-1)}, j) = \alpha(\mathcal{L}_{(2k_u)}, i) = \alpha(\mathcal{L}_{(2k_u)}, j)$ である．また，特に，$\mathcal{L}_{(2k_u-1)}[\alpha(\mathcal{L}_{(2k_u-1)}, i)]$, $\mathcal{L}_{(2k_u)}[\alpha(\mathcal{L}_{(2k_u)}, i)]$ は，i, j を同時に含むハッシュ連鎖の中で，その長さが最も短いものである．

このとき，ユーザ i, j は，いずれも以下の 4 個のハッシュ値を計算できる:

$$\begin{aligned} \nu_1 &:= C_N^I(s_1)[j], \\ \nu_2 &:= C_N^{II}(s_2)[i], \\ \nu_3 &:= (\mathcal{L}_{(2k_u-1)}[\alpha(\mathcal{L}_{(2k_u-1)}, i)])[i - \beta_{i,2k_u-1}], \\ \nu_4 &:= (\mathcal{L}_{(2k_u)}[\alpha(\mathcal{L}_{(2k_u)}, i)])[j - \beta_{j,2k_u}] \end{aligned} \tag{8}$$

ここで，$\beta_{i,j} := (N \cdot (\alpha(\mathcal{L}_{(j)}, i) - 1))/k_{\mathcal{L}_{(j)}}$ とおいた．こうして，ユーザ i, j の共通鍵 $K_{i,j}$ を，

$$K_{i,j} := F(\nu_1, \nu_2, \nu_3, \nu_4) . \tag{9}$$

と計算すればよい．

2.7 セキュリティ評価

ハッシュ連鎖アグリゲーションを用いた共通鍵作成法に対して，攻撃者が一人の場合と，二人の場合について，次のようにセキュリティが評価されている．

攻撃者が一人の場合は，定理 1 を証明することができる．

定理 1. 一人の攻撃者は，任意の i, j $(1 \le i < j \le N)$ について，ユーザ i, j の共通鍵 $K_{i,j}$ を計算できない．

証明は紙面の制約上，省略する．[13] を参照せよ．

2 人以上の攻撃者が結託する場合は，定理 2 を証明することができる．

定理 2. ユーザ i, j $(1 \le i < j \le N)$ について，$i < a_1 \le N(\alpha - 1)/k + N/2k$, $N(\alpha - 1)/k + N/2k + 1 \le a_2 < j$ を満たすユーザ a_1, a_2 は，結託してユーザ i, j の共通鍵 $K_{i,j}$

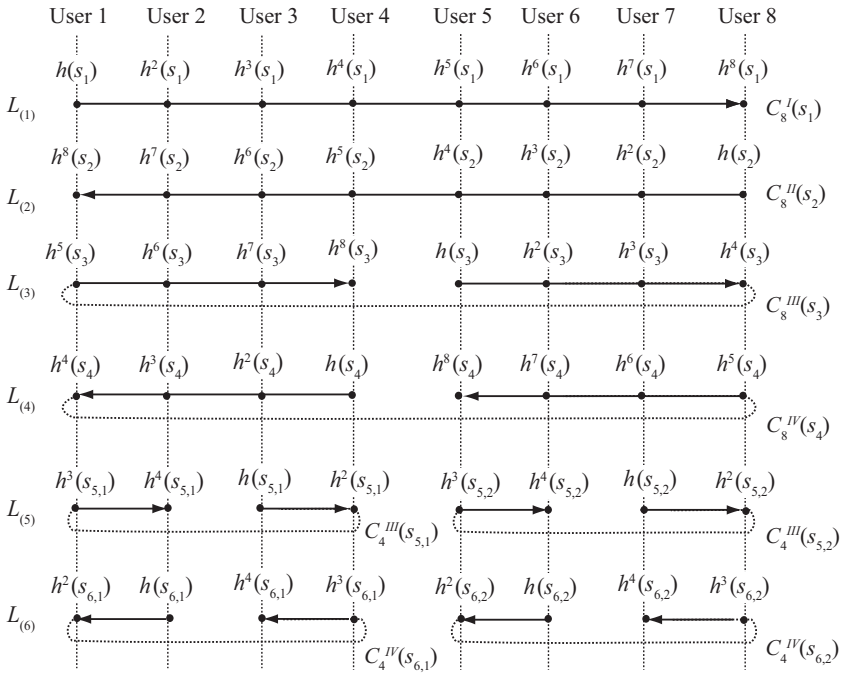

図 1 $N = 8$ のときの $\mathcal{HCA}$
Fig. 1 Our $\mathcal{HCA}$ ($N = 8$)

を計算することができる．ここで，α, k の定義は定理 *1* のものと同じである．

証明は紙面の制約上，省略する．[13] を参照せよ．

ハッシュ連鎖アグリゲーションを用いた共通鍵作成法では，2 人以上のユーザが結託すると，別の二人のユーザの共通鍵を計算できる状況がある．

3. 結託攻撃への対策法

本節では，ハッシュ連鎖アグリゲーションを用いた共通鍵作成法における，結託攻撃に対する対策法を提案する．

3.1 multiple-$\mathcal{HCA}$

結託攻撃への対策のために，$\mathcal{HCA}$ を複数使用することを考える．定理 2 で述べたように，ある $\mathcal{HCA}$ において，ある 2 ユーザ i, j の共通鍵を，攻撃者 a_1, a_2 の結託によって計算可能であるとする．このとき，攻撃者 a_1, a_2 の結託によって，ユーザ i, j の共通鍵を計算出来ないような $\mathcal{HCA}$ を用意し，その $\mathcal{HCA}$ のハッシュ値を各ユーザに，追加で割り当てる．2 つの $\mathcal{HCA}$ を用いて共通鍵の作成を行うことで，片方の $\mathcal{HCA}$ で結託攻撃が成立しても，もう一方の $\mathcal{HCA}$ では攻撃が成立しないため，攻撃者 a_1, a_2 の結託による結託攻撃を防ぐことが可能である．また，ユーザ i, j が，どちらの $\mathcal{HCA}$ でも共通鍵を作成可能であることは自明である．これが，multiple-$\mathcal{HCA}$ の基本的なアイディアである．multiple-$\mathcal{HCA}$ の概念図を図 2 に示す．

以下，具体的にどのように multiple-$\mathcal{HCA}$ を実現するかを説明する．

まず，一つ目の $\mathcal{HCA}$ を $\mathcal{HCA}_1$ とする．次に，ユーザの配置のみを任意に並び替えた $\mathcal{HCA}$ を作成し，$\mathcal{HCA}_2$ とする．ユーザの配置のみを並び替えるとは，$\mathcal{HCA}_1$ において，ユーザ j が配置されていた場所に，$\mathcal{HCA}_2$ では，ユーザ i が配置されるとすると，ユーザ i は，$V(\mathcal{HCA}_2, j)$ を割り当てられ，$\mathcal{HCA}_1$ のハッシュ値に加えて保持するということである．このとき，$\mathcal{HCA}_1$ におけるユーザ j の保持する $V(\mathcal{HCA}_1, j)$ とは異なるハッシュ値を保持することに注意されたい．[*1]ここで，$\mathcal{HCA}_1$ の各ユーザ id を，multiple-$\mathcal{HCA}$ のパスと呼ぶこととし，上記の例の場合，$\mathcal{HCA}_2$ において，ユーザ i はパス j に配置されたと言う．すなわち，$\mathcal{HCA}_k$ において，パス j に配置されるユーザ i は，$V(\mathcal{HCA}_k, j)$ を保持するということである．

任意にユーザの配置のみを並び替えた $\mathcal{HCA}$ では，あるユーザ i, j に対して，結託攻撃可能な攻撃者の集合はそれぞれ異なるため，全ての $\mathcal{HCA}$ の結託攻撃可能な攻撃者の集合に含まれる 2 ユーザ（攻撃者）にのみ，結託攻撃が可能である．すなわち，$\mathcal{HCA}$ が増えるほど，結託攻撃のリスクは小さくなると考えられる．

3.2 各ユーザの保持するハッシュ値の個数

multiple-$\mathcal{HCA}$ において，$\mathcal{HCA}$ の個数が増えるほど，結託攻撃に対するリスクが減少することを 3.1 節にて述べた．理論的には，ユーザ数を N とすると，ユーザの並び替えのパターンは $N!$ 通りであるため，$N!$ 個の $\mathcal{HCA}$ によって，

[*1] すなわち，$\mathcal{HCA}_1$ と $\mathcal{HCA}_2$ は異なる種を用いる．

Vol.2018-DPS-174 No.22
Vol.2018-CSEC-80 No.22
2018/3/6

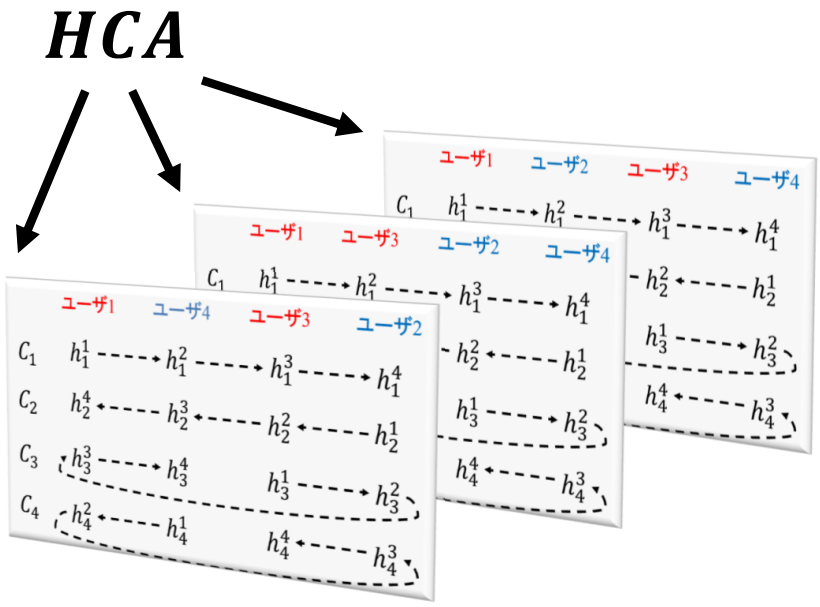

図 2　multiple-$\mathcal{HCA}$
Fig. 2　multiple-$\mathcal{HCA}$

結託攻撃を完全に防ぐことが出来る．しかし，各ユーザが保持するハッシュ値の個数も増加するため，出来るだけ少ない $\mathcal{HCA}$ によって，結託攻撃のリスクを減らすことを考えなければならない．次節にて，より少ない $\mathcal{HCA}$ で，結託攻撃のリスクを減らすユーザの配置方式を提案する．

3.3　ユーザ配置方式の提案

ユーザ配置方式の提案のために，まずは $N = 4$ の $\mathcal{HCA}$ で考える．$N = 4$ の $\mathcal{HCA}$ では，表 1 に示す，4 つの結託攻撃パターンが存在する．

表 1　$N = 4$ の $\mathcal{HCA}$ の結託攻撃パターン

認証ユーザ	攻撃者
ユーザ 1, 3	ユーザ 2, 4
ユーザ 2, 4	ユーザ 1, 3
ユーザ 1, 4	ユーザ 2, 3
ユーザ 2, 3	ユーザ 1, 4

上記の結託攻撃パターンを完全に防ぐためのユーザの並びは次の通りである．

表 2　結託攻撃を防ぐための $\mathcal{HCA}$ のユーザ配置（ $N = 4$ ）

パス	1	2	3	4
$\mathcal{HCA}_1$	1	2	3	4
$\mathcal{HCA}_2$	1	3	2	4
$\mathcal{HCA}_3$	1	4	3	2

表 2 に示したユーザの配置の 3 つの $\mathcal{HCA}$ は，
表 1 の攻撃パターンを全て防ぐための，必要最小限の multiple-$\mathcal{HCA}$ である．

このように，$N = 4$ の $\mathcal{HCA}$ において，パス 1 と 2，パス 3 と 4 のいずれかに，全てのユーザのペアを配置することで，結託攻撃を防ぐことが出来る．

これを，$N = 8$ に拡張する．方針としては，長さ 4 のハッシュ連鎖の 1 番目と 2 番目及び 3 番目と 4 番目に，全てのユーザのペアを割り当てるようにユーザを配置する．$N = 4$ のときと同様に，長さ 4 のハッシュ連鎖の 1 番目と 2 番目及び 3 番目と 4 番目に割り当てられたユーザのペアに対しては，結託攻撃が成立しない．

実際に，$N = 8$ に拡張したものを表 3 に示す．

表 3　結託攻撃を防ぐための $\mathcal{HCA}$ のユーザ配置（ $N = 8$ ）

パス	1	2	3	4	5	6	7	8
$\mathcal{HCA}_1$	1	2	3	4	5	6	7	8
$\mathcal{HCA}_2$	1	3	2	4	5	7	6	8
$\mathcal{HCA}_3$	1	4	3	2	5	8	7	6
$\mathcal{HCA}_4$	1	5	2	6	3	7	4	8
$\mathcal{HCA}_5$	1	6	5	2	3	8	7	4
$\mathcal{HCA}_6$	1	7	2	8	5	3	6	4
$\mathcal{HCA}_7$	1	8	7	2	5	4	3	6

さらに，同様にして，$N = 16$ に拡張したものを表 4 に示す．

3.4　必要な $\mathcal{HCA}$ の個数

multiple-$\mathcal{HCA}$ において，$N!$ 個の $\mathcal{HCA}$ があれば，結託攻撃を完全に防ぐことが出来ることは 3.2 節で書いた．本節では，multiple-$\mathcal{HCA}$ において，結託攻撃を完全に防ぐために必要となる $\mathcal{HCA}$ の個数を求める．

表 4　結託攻撃を防ぐための $\mathcal{HCA}$ のユーザ配置（ $N = 16$ ）

パス	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\mathcal{HCA}_1$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\mathcal{HCA}_2$	1	3	2	4	5	7	6	8	9	11	10	12	13	15	14	16
$\mathcal{HCA}_3$	1	4	3	2	5	8	7	6	9	12	11	10	13	16	15	14
$\mathcal{HCA}_4$	1	5	2	6	3	7	4	8	9	13	10	14	11	15	12	16
$\mathcal{HCA}_5$	1	6	5	2	3	8	7	4	9	14	13	10	11	16	15	12
$\mathcal{HCA}_6$	1	7	2	8	5	3	6	4	9	15	10	16	13	11	14	12
$\mathcal{HCA}_7$	1	8	7	2	5	4	3	6	9	16	15	10	13	12	11	14
$\mathcal{HCA}_8$	1	9	2	10	3	11	4	12	5	13	6	14	7	15	8	16
$\mathcal{HCA}_9$	1	10	9	2	3	12	11	4	5	14	13	6	7	16	15	8
$\mathcal{HCA}_{10}$	1	11	2	12	9	3	10	4	5	15	6	16	7	13	8	14
$\mathcal{HCA}_{11}$	1	12	11	2	9	4	3	10	5	16	15	6	7	14	13	8
$\mathcal{HCA}_{12}$	1	13	2	14	3	15	4	16	9	5	10	6	11	7	12	8
$\mathcal{HCA}_{13}$	1	14	13	2	3	16	15	4	9	6	5	10	11	8	7	12
$\mathcal{HCA}_{14}$	1	15	2	16	13	3	14	4	9	7	10	8	5	11	6	12
$\mathcal{HCA}_{15}$	1	16	15	2	13	4	3	14	9	8	7	10	5	12	11	6

3.3 節の方式では，存在し得る全てのユーザのペアについて，それぞれ一度ずつ，ペアを並べて配置する．すなわち，ユーザ数 N に対して，$\binom{N}{2}$ 通り存在するユーザのペアを並び替えることを考えれば良い．1 つの $\mathcal{HCA}$ につき，$N/2$ 個のユーザのペアを配置することが出来るため，必要な $\mathcal{HCA}$ の個数 K は以下のように計算出来る．

$$\begin{aligned} K &= \frac{\binom{N}{2}}{\frac{N}{2}} \\ &= N - 1 \end{aligned}$$

よって，ユーザ数 N の multiple-$\mathcal{HCA}$ に対して，$N-1$ 個の $\mathcal{HCA}$ があれば結託攻撃を完全に防ぐことが出来る．また，ユーザ数 N の multiple-$\mathcal{HCA}$ に対して，K 個の $\mathcal{HCA}$ を使用するとき，結託攻撃を受ける可能性のあるユーザのペア数は，

$$\binom{N}{2} - \frac{N}{2} \times K = \frac{N\{(N-1) - K\}}{2}$$

となる．

上記の通り，3.3 節の方式では，結託攻撃の危険性を完全に無くすために，$N-1$ 個の $\mathcal{HCA}$ を必要とする．しかし，あらかじめ信頼出来るユーザが複数いる場合には，ユーザの配置を上手く工夫することにより，少なくとも，信頼出来るユーザ同士の認証については，より少ない $\mathcal{HCA}$ で，安全に認証を行うことが出来る．例えば，$N = 8$ において，ユーザ 1, 2, 5, 6 が信頼出来るユーザ同士であるとわかっている場合には，表 5 に示すような $\mathcal{HCA}$ を用意すれば良い．

5 に示した $\mathcal{HCA}$ によって，ユーザ 1, 2, 5, 6 同士の認証であれば結託攻撃は成立しない．しかし，ユーザ 1, 2, 5, 6 とその他のユーザの認証では，結託攻撃が成立する場合がある．multiple-$\mathcal{HCA}$ は，結託攻撃の危険性を完全に無くすために，$N-1$ 個の $\mathcal{HCA}$ を必要とする．これは，あくまで結託攻撃の危険性を完全に無くすために必要な個数である．すなわち，ある特定のユーザの認証についてのみを考えれば，より少ない $\mathcal{HCA}$ で良いため，信頼出来るユーザかつ優先的に保護したいユーザを上手く配置して，より少ない $\mathcal{HCA}$ によって運用されることが望ましい．

表 5　ユーザ 1, 2, 5, 6 が信頼出来るときの $\mathcal{HCA}$ のユーザ配置の例（ $N = 8$ ）

パス	1	2	3	4	5	6	7	8
$\mathcal{HCA}_1$	1	2	5	6	3	4	7	8

また，攻撃者である可能性が高いユーザが存在する場合にも，ユーザの配置を工夫することで，より少ない $\mathcal{HCA}$ によって，結託攻撃の危険性を完全に無くすことが出来る．ユーザ 3, 5, 7 が攻撃者である可能性が高いとき，表 6 のように multiple-$\mathcal{HCA}$ を構成すれば，ユーザ 3, 5, 7 が結託しても，その他のユーザへの結託攻撃は成立しない．

表 6　ユーザ 3, 5, 7 が攻撃者である可能性が高いときの $\mathcal{HCA}$ のユーザ配置の例（ $N = 8$ ）

パス	1	2	3	4	5	6	7	8
$\mathcal{HCA}_1$	1	2	4	6	8	3	5	7

3.5　3 人以上のユーザによる認証

本節では，ハッシュ連鎖アグリゲーション及び multiple-$\mathcal{HCA}$ を用いて，3 人以上のユーザ同士で共通鍵を生成する方法について議論する．まず，$N = 4$ の $\mathcal{HCA}$ について考える．$N = 4$ の $\mathcal{HCA}$ では，以下のようにハッシュ値を

計算することで，3 人以上の任意のグループの共通鍵を作成することが出来る．

ユーザ 1, 2, 3, 4 のグループ　グループ内の全てのユーザが $h^4(s_1), h^4(s_2)$ を計算可能である．
よって，ハッシュ値 $h^4(s_1), h^4(s_2)$ を用いてユーザ 1, 2, 3, 4 の共通鍵を作成する．

ユーザ 1, 2, 3 のグループ　グループ内の全てのユーザが $h^3(s_1), h^4(s_2)$ を計算可能である．
しかし，ユーザ 4 からは $h^3(s_1)$ を計算出来ない．
よって，ハッシュ値 $h^3(s_1), h^4(s_2)$ を用いてユーザ 1, 2, 3 の共通鍵を作成出来る．

ユーザ 1, 2, 4 のグループ　グループ内の全てのユーザが $h^4(s_3), h^3(s_4)$ を計算可能である．
しかし，ユーザ 3 からは $h^3(s_4)$ を計算出来ない．
よって，ハッシュ値 $h^4(s_3), h^3(s_4)$ を用いてユーザ 1, 2, 4 の共通鍵を作成出来る．

ユーザ 1, 3, 4 のグループ　グループ内の全てのユーザが $h^3(s_3), h^4(s_4)$ を計算可能である．
しかし，ユーザ 2 からは $h^3(s_3)$ を計算出来ない．
よって，ハッシュ値 $h^3(s_3), h^4(s_4)$ を用いてユーザ 1, 3, 4 の共通鍵を作成出来る．

ユーザ 2, 3, 4 のグループ　グループ内の全てのユーザが $h^4(s_1), h^3(s_2)$ を計算可能である．
しかし，ユーザ 1 からは $h^3(s_2)$ を計算出来ない．
よって，ハッシュ値 $h^4(s_1), h^3(s_2)$ を用いてユーザ 2, 3, 4 の共通鍵を作成出来る．

上記のように，$N = 4$ の $\mathcal{HCA}$ では，グループ内の任意のユーザの組み合わせに対して，共通鍵を作成可能である．

次に，$N = 8$ に拡張することを考える．$N = 4$ の $\mathcal{HCA}$ では，グループ内の任意のユーザの組み合わせに対して，共通鍵を作成可能であることから，長さ 4 の同じハッシュ連鎖に割り当てられたユーザ同士は，そのグループ内の任意のユーザの組み合わせに対して，共通鍵を作成可能であると言える．例えば，図 1 の $\mathcal{HCA}$ では，ユーザ 1, 2, 3, 4 のグループとユーザ 5, 6, 7, 8 のグループは，$N = 4$ のときと同様にして，それぞれのグループ内で任意の共通鍵を作成できる．

このことから，存在し得る 4 人のユーザの組み合わせを，長さ 4 のハッシュ連鎖に割り当てるように multiple-$\mathcal{HCA}$ を構築することで，multiple-$\mathcal{HCA}$ 中のいずれかの $\mathcal{HCA}$ を用いて，4 人以下の任意のユーザのグループの共通鍵を作成できる．しかし，全ての 4 人以下のユーザのグループに対して，これを実装するために必要な $\mathcal{HCA}$ の個数は，

$$\frac{\binom{8}{4}}{\frac{8}{4}} = 35$$

ユーザ数が N の場合には，

$$\frac{\binom{N}{4}}{\frac{N}{4}}$$

となる．

よって，3 人以上のユーザの認証についても，信頼出来るユーザや，3 人以上で認証する可能性の高いユーザ等の条件を踏まえた上で，適切にユーザを配置し，実装されることが望ましい．

3.6 ユーザ数が動的に変化する場合

本節では，ユーザ数が動的に変化する場合について議論する．基本的には，ダミーユーザを予め KDC が作成し，ダミーユーザに割り当てられたハッシュ値を，グループに追加されるユーザに割り当てる．また，ユーザをグループから削除する際には，そのユーザの持つハッシュ値を削除すればよい．尚，ここでは，ユーザ数の上限は変化しないものとする．$N = 8$ の場合について，表 7 のような multiple-$\mathcal{HCA}$ を考える．

表 7　ユーザ数が変化する際の $\mathcal{HCA}$ のユーザ初期配置の例（$N = 8$）

パス	1	2	3	4	5	6	7	8
$\mathcal{HCA}_1$	A	B	C	D	*	*	*	*

表 7 において，A, B, C, D はそれぞれユーザであり，* にはまだユーザが割り当てられていないがダミーユーザとして KDC がハッシュ値を割り当てているものとする．また，このときユーザ A,B,C,D のグループは，任意のユーザの組み合わせに対して，共通鍵を作成可能である．

ここで，新たなユーザ E が追加されるとき，multiple-$\mathcal{HCA}$ は以下のようになる．

表 8　ユーザ E が追加されたときの $\mathcal{HCA}$ のユーザ配置（$N = 8$）

パス	1	2	3	4	5	6	7	8
$\mathcal{HCA}_1$	A	B	C	D	E	*	*	*

表 8 において，ユーザ E には，KDC から個別にハッシュ値を割り当てられている．また，このときユーザ A,B,C,D,E のグループにおいて，A, B, C, D, E のユーザの組み合わせに対してのみ，グループの共通鍵を作成可能であり，ユーザ A,B,C,D 内で認証が行われる際，ユーザ E からはいかなる攻撃も成立しない．

このように，ユーザの初期配置によって，より保護したいユーザや危険度の高いユーザに対応することが出来る．

次に，ユーザを削除する際は以下のようになる．例として，ユーザ B を削除するとき，表 9 のようになる．

このように，ユーザを削除した後に，ユーザの配置を並

表 9 ユーザ B が削除されたときの $\mathcal{HCA}$ のユーザ配置（$N = 8$）

パス	1	2	3	4	5	6	7	8
$\mathcal{HCA}_1$	A	*	C	D	E	*	*	*

び替えることは出来ない．ユーザ A, B, C, D は，より強く保護出来ていたため，ユーザ B が割り当てられていた，パス 2 には，信頼出来るユーザを配置することが望ましい．

4. 結論

本論文では，ハッシュ連鎖アグリゲーションを用いた共通鍵作成法における結託攻撃への対策について議論した．結託攻撃への対策法である multiple-$\mathcal{HCA}$ は，$\mathcal{HCA}$ を複数使用するという発想によって，結託攻撃への対策に成功しただけでなく，3 人以上の認証や，ネットワークのユーザ数が動的に変化する場合について議論し，$\mathcal{HCA}$ のさらなる可能性を切り開いた．

今後の課題は，multiple-$\mathcal{HCA}$ に対するさらなる解析である．

謝辞

本研究は科学研究費補助金 (課題番号 JP15K00189) の助成，および国立研究開発法人科学技術振興機構（JST）の国際科学技術協力基盤整備事業の支援を受けたものである．

参考文献

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] J. Katz and Y. Lindell, *Introduction to Modern Cryptography: Principles and Protocols*. Chapman and Hall/CRC, 2007.

[3] D. J. Bernstein, J. Buchmann, and E. Dahmen, Eds., *Post-Quantum Cryptography*. Springer-Verlag, 2009.

[4] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[5] R. Dutta, E. Chang, and S. Mukhopadhyay, “Efficient self-healing key distribution with revocation for wireless sensor networks using one way key chains,” in *Proceedings of 5th International Conference on Applied Cryptography and Network Security (ACNS)*, ser. Lecture Notes in Computer Science, no. 4521. Springer-Verlag, 2007, pp. 385–400.

[6] M. Joye and S. Yen, “One-way cross-trees and their applications,” in *Public Key Cryptography (PKC)*, ser. Lecture Notes in Computer Science, vol. 2274. Springer-Verlag, 2002, pp. 346–356.

[7] L. Lamport, “Password authentication with insecure communication,” vol. 24, no. 11, pp. 770–772, Nov. 1981.

[8] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, “Efficient authentication and signing of multicast streams over lossy channels,” in *Proceedings of the IEEE Symposium on Security and Privacy*, May 2000, pp. 56–73.

[9] R. Rivest and A. Shamir, “PayWord and MicroMint: Two simple micropayment schemes,” in *Security Protocols*, ser. Lecture Notes in Computer Science, vol. 1189. Springer-Verlag, 1997, pp. 69–87.

[10] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Oct. 2008.

[11] X. Guo, S. Huang, L. Nazhandali, and P. Schaumont, “Fair and comprehensive performance evaluation of 14 second round SHA-3 ASIC implementations,” NIST 2nd SHA-3 Candidate Conference, Aug. 2010.

[12] D. Coppersmith and M. Jakobsson, “Almost optimal hash sequence traversal,” in *Proceesings of 6th International Conference on Financial Cryptography (FC 2002)*, ser. Lecture Notes in Computer Science, vol. 2357. Springer-Verlag, 2002, pp. 102–119.

[13] Y. Kurihara and M. Soshi, “A Novel Hash Chain Construction for Simple and Efficient Authentication,” Privacy, Security and Trust, Dec.2016.