Computer Security Symposium 2019
21 - 24 October 2019

ハッシュチェーン計算によるモデル化

平井 晨太[1,a)]　双紙 正和[1]

概要：IoT 機器の普及により，メモリ等に制約のある機器が増加した．こうした IoT 機器同士で安全な通信を行うには，ユーザ数を N としたとき，N 個の鍵を持てばよい．しかし，その場合ユーザ数の増加に伴い必要な鍵の数も莫大になる．そこで軽量で効率的に相互認証を行う方法として，ハッシュチェーンアグリゲーション (以下 HCA) が栗原によって考案された [1]．HCA は，各ユーザにハッシュ値を複数割り当て，それらを利用して共通鍵を生成する方法であり，各ユーザが保持する必要のあるハッシュ値は $O(\log N)$ 個である．しかし，HCA には複数人が結託して攻撃するとき，認証者以外が認証者同士の共通鍵を計算することができる場合がある．そこで，栗原の HCA を改良した HCA[2] が提案された．しかし，これは栗原の HCA よりも各ユーザが保持する必要があるハッシュ値の数が増加している．本稿では，両方のハッシュチェーンアグリゲーションを表現できる，より一般的なハッシュチェーンアグリゲーションを提案し，さらに，ハッシュチェーン計算モデルを用いて共通鍵生成に必要なハッシュ値について述べる．

キーワード：IoT, 認証, ハッシュ関数, ハッシュチェーン

Hash chain computation models

Shinta Hirai[1,a)]　Masakazu Soshi[1]

Abstract: With the spread of IoT devices, the number of devices with limited memory has increased. To perform secure communication between IoT devices, if you have N users, you only need to have N keys. However, in that case, the number of necessary keys becomes enormous as the number of users increases. Therefore, hash chain aggregation (HCA) was devised by Kurihara as a lightweight and efficient mutual authentication method. HCA is a method of assigning multiple hash values to each user and using them to generate a common key. Each user needs to hold $O(\log N)$ hash values. However, in HCA, when multiple people collide and attack, non-certifiers may be able to calculate a common key between authenticators. Therefore, an HCA with an improved Kurihara's HCA has been proposed, but this requires an increased number of hash values for each user than the Kurihara's HCA. In this paper, we propose a more general hash chain aggregation that can express both hash chain aggregations, and describe the hash values necessary for common key generation using a hash chain calculation model.

1. はじめに

IoT 機器の普及により，IoT 機器同士でネットワークを形成して相互に通信を行う機会が増加した．ユーザ数を N としたとき，各ユーザが N 個の鍵を持てば安全に相互認証を行うことができるが，ユーザ数が増加すると各ユーザが保持する必要のある鍵は莫大になる．しかし IoT 機器はコストの削減や省スペース化のため，計算能力やメモリ等に制約のあるものが多く，こうした機器同士でも軽量で安全な相互認証を行うことは重要である．

軽量に相互認証を行う方法の一つとしてハッシュチェーンアグリゲーション (以下 HCA) を用いた方法がある [1]．HCA はハッシュチェーンを複数用いて計算したハッシュ値をネットワークに接続された各ユーザに配布する．それぞれのユーザが通信をする際には，配布されたハッシュ値を用いて共通鍵を生成し相互認証を行うという方法である．この方法では各ユーザが保持するハッシュ値は $O(\log N)$

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

個のみであり，ハッシュ関数を用いているので共通鍵の生成に必要な計算が少ない．しかし，この方法では相互認証をする際，攻撃者は一人であると仮定しており，複数の攻撃者が結託して攻撃する場合は正しく相互認証を行えないことがある．そこで，各ユーザが保持するハッシュ値を$O(\log N)$個としたまま，結託攻撃が成功するパターンが少なくなるような改良を行った HCA が提案された [2]．しかし，改良した HCA では，栗原の HCA に新たなハッシュチェーンを追加しており，各ユーザが保持する必要のあるハッシュ値は$O(\log N)$ではあるが，栗原の HCA よりも増加している．よって我々は，両方のハッシュチェーンアグリゲーションを表現できる，より一般的なハッシュチェーンアグリゲーションを提案するとともに，ハッシュチェーン計算モデルを用いて共通鍵生成に必要なハッシュ値について述べる．本論文は以下のように構成される．第 2 節ではより一般的なハッシュチェーンアグリゲーションの構成法と鍵の生成方法について説明し，第 3 節では，共通鍵生成に必要なハッシュ値の個数について述べる．最後にまとめを述べる．

2. 提案手法

2.1 ハッシュチェーンの定義

本章では，ハッシュチェーンアグリゲーション及び，それを用いた共通鍵生成法について説明する．まずはハッシュ関数 h，種 s を用いて，ハッシュチェーンを以下のように表す．$h^1(s) := h(s)$, $h^i(s) := h(h^{i-1}(s))$ ただし，$(i \geq 2)$ とする．整数 n に対して $(i_1, i_2, \ldots, i_n)$ は集合 $\{1, 2, \ldots, n\}$ の順列とし，種 s を持つ長さ n のハッシュチェーンは次の通りに定義する．

$$(v_1, v_2, \ldots, v_n) \quad \text{where } v_j = h^{i_j}(s_j), (1 \leq j \leq n) \tag{1}$$

n 個のハッシュ値は $h(s)$, $h^2(s)$, …, $h^n(s)$ の順で計算することができる．

ハッシュチェーンアグリゲーションの基本的な考え方は以下の通りである．まず，Key Distribution Center (KDC) と N 人のユーザが存在する．KDC は安全で信頼できる方法で各ユーザと通信することができる．KDC は各ユーザに配布するハッシュ値を事前に計算し，各ユーザに配布する．以降，各ユーザ同士で通信する際には配布されたハッシュ値を，ハッシュ関数などの一方向性関数 F に適用し，共通鍵を生成して相互認証を行う．

2.2 基本的なハッシュチェーン

ユーザ数を N としたときの，HCA における一般的なハッシュチェーンについて述べる．

以降では，簡潔にするため，いくつかの整数 m (≤ 2) について，$N = 2^m$ と仮定する．基本的なハッシュチェーンを定義する．ある正整数 b について，ハッシュチェーンの長さを $\ell = 2^b$ $(1 \leq b \leq m)$ と仮定し，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 をハッシュチェーンの seed とする．

$$C_\ell(s) := (h^1(s), h^2(s), \ldots, h^\ell(s)) \tag{2}$$

また，以降では，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．

2.3 ハッシュチェーンリストの定義

次にハッシュチェーンリストを以下のように定義する．

$$\mathcal{L}(k, s_1, \ldots, s_k) := \{C_{N/k}(s_1), C_{N/k}(s_2), \ldots, C_{N/k}(s_k)\} \tag{3}$$

ここで正整数 f $(0 \leq f \leq m-2)$ を用いて $k = 2^f$ と表し，$s_1, s_2, ..., s_k$ を種の列とする．また，あるハッシュチェーンリストに含まれる各ハッシュチェーンの長さは等しいとする．

以降では簡単化のために $\mathcal{L}(k, s_1, \ldots, s_k)$ の k, s_1, …, s_k を省略して単純に $\mathcal{L}$ とする．またハッシュチェーンリスト $\mathcal{L}$ が与えられたとき，それに属するハッシュチェーンの数を $k_\mathcal{L}$ と表す．

2.4 ハッシュチェーンアグリゲーションの定義

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

$$\mathcal{A} := \{\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_r\} \tag{4}$$

また，$\mathcal{L}_i$ $(1 \leq i \leq r)$ は，式 (3) によって次のように表すことができる．

$$\mathcal{L}_i := \{C_{N/k_{\mathcal{L}_i}}(s_{i,1}), C_{N/k_{\mathcal{L}_i}}(s_{i,2}), ..., C_{N/k_{\mathcal{L}_i}}(s_{i,k_{\mathcal{L}_i}})\} \tag{5}$$

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

2.5 ユーザに配布されるハッシュ値

ハッシュ値 $h^i(s_j)$ から，それを保持しているユーザ u がわかるような関数 $\mathcal{U}$ があるとする．

$$\mathcal{U}(h^i(s_j)) := u \tag{6}$$

すると，あるハッシュチェーンに含まれるハッシュ値を持つユーザの集合は，関数 P を用いて次のように表すことができる．

$$P(C_\ell(s_j)) := \{\mathcal{U}(h^i(s_j)) \mid h^i_j \in C_\ell(s_j)\} \tag{7}$$

さらに，あるユーザが所属するハッシュチェーンの集合を，

Γ を用いて次のように表すことができる．

$$\Gamma(u) := \{C_\ell(s_j) \mid u \in P(C_\ell(s_j))\} \tag{8}$$

また，あるユーザが，ハッシュチェーンアグリゲーションによって相互認証を行うことができるユーザの集合は，

$$\{P(C_\ell(s_j)) \mid C_\ell \in \Gamma(u)\} \tag{9}$$

となる．この集合の要素数が，ハッシュチェーンアグリゲーションに参加しているユーザ数 N よりも少ない場合，ハッシュチェーンアグリゲーションを用いて相互認証を行うことができないユーザの組み合わせが存在する．

2.6 ハッシュチェーンアグリゲーションによる共通鍵生成

ユーザ p,q による，ハッシュチェーンアグリゲーションを用いた相互認証のための共通鍵作成法は以下のとおりである．

ユーザ p,q がそれぞれ所属するハッシュチェーンの集合において，両方が所属するハッシュチェーンの集合は，$\Gamma(p) \cup \Gamma(q)$ である．すると，共通鍵作成に必要なハッシュ値の集合は以下の通りになる

$$\{h_j^i \mid h_j^i \in \{\Gamma(p) \cap \Gamma(q)\}, \mathcal{U}(h^i(s_j)) = \{p \cup q\}\} \tag{10}$$

最後にこの集合に含まれるハッシュ値を一方向性関数 F に入力することで，ユーザ p,q の共通鍵 Kp,q を作成することができる．

2.7 栗原の HCA による例

例として，$N = 8$ の場合の栗原のハッシュチェーンアグリゲーションにおいて，提案手法を用いてユーザ $3,6$ が共通鍵を生成する場合について述べる．

図 1 より，$\Gamma(3) = \{C_8^I(s_1), C_8^{II}(s_2), C_8^{III}(s_3), C_8^{IV}(s_4), C_4^{III}(s_{5,1}), C_4^{IV}(s_{6,1})\}$ であり，$\Gamma(6) = \{C_8^I(s_1), C_8^{II}(s_2), C_8^{III}(s_3), C_8^{IV}(s_4), C_4^{III}(s_{5,2}), C_4^{IV}(s_{6,2})\}$ である．

両方が所属するハッシュチェーンの集合は $\{\Gamma(3) \cup \Gamma(6)\} = \{C_8^I(s_1), C_8^{II}(s_2), C_8^{III}(s_3), C_8^{IV}(s_4)\ \}$ である．

これらのハッシュチェーンに含まれるユーザ $3,6$ がともに計算可能なハッシュ値は $\{h^3(s_1), h^3(s_2), h^2(s_3), h^2(s_4)\ \}$ であり，これらのハッシュ値を一方向性関数 F に入力することで，ユーザ $3,6$ の共通鍵 $K_{3,6}$ を作成することができる．

2.8 平井の HCA による例

次は栗原のハッシュチェーンアグリゲーションを改良したハッシュチェーンアグリゲーションにおいて，提案手法を用いてユーザ $2,7$ が共通鍵を生成する場合について述べる．

図 2 より，$\Gamma(2) = \{C_8^{S1}(s_1), C_8^{S2}(s_2), C_8^{S3}(s_3), C_8^{S4}(s_4), C_4^{S1}(s_{5,2}),\ C_4^{S2}(s_{6,2}),\ C_4^{S3}(s_{7,1}),\ C_4^{S4}(s_{8,1})\}$ であり，$\Gamma(7) = \{C_8^{S1}(s_1), C_8^{S2}(s_2), C_8^{S3}(s_3), C_8^{S4}(s_4), C_4^{S1}(s_{5,2}), C_4^{S2}(s_{6,2}), C_4^{S3}(s_{7,2}), C_4^{S4}(s_{8,2})\}$ である．

両方が所属するハッシュチェーンの集合は $\{\Gamma(2) \cup \Gamma(7)\} = \{C_8^{S1}(s_1), C_8^{S2}(s_2), C_8^{S3}(s_3), C_8^{S4}(s_4), C_4^{S1}(s_{5,2}), C_4^{S2}(s_{6,2})\ \}$ である．

これらのハッシュチェーンに含まれるユーザ $2,7$ がともに計算可能なハッシュ値は $\{h^7(s_1), h^7(s_2), h^6(s_3), h^6(s_4)\ h^3(s_{5,2}), h^3(s_{6,2})\ \}$ であり，これらのハッシュ値を一方向性関数 F に入力することで，ユーザ $2,7$ の共通鍵 $K_{2,7}$ を作成することができる．

3. 共通鍵生成に必要な最少のハッシュ値の数

前節で述べた共通鍵生成法において，共通鍵生成に用いたハッシュ値のうち，認証者は計算可能であるが攻撃者は計算できないといったハッシュ値が複数含まれている場合がある．しかし，実際は認証者は計算可能であるが攻撃者は計算できないハッシュ値がたった一つあるだけで共通鍵を生成することができる．そこで本節ではそういったハッシュ値の数が最少 (4 個より少ない) となるような共通鍵が生成できることを示す．結託攻撃を行うユーザを a,b，認証を行いたいユーザを p,q とする．また，以降ではユーザ k を U_k と表す．KDC によって，U_k に配布されるハッシュ値の集合を B_k をすると，U_k が計算可能なハッシュ値の集合は以下のように表すことができる．

- U_k が U_a または U_b の時

$$\Theta_k := \{h_j^m \mid h_j^i \in \{B_a \cup B_b\}, i \le m \le \ell\} \tag{11}$$

- 上記以外の時

$$\Theta_k := \{h_j^m \mid h_j^i \in B_k, i \le m \le \ell\} \tag{12}$$

ここで，U_p, U_q は計算可能であり，攻撃者 U_a, U_b は計算できないハッシュ値を θ とすると．

$$\theta = (\Theta_p \cap \Theta_q) - (\Theta_a \cap \Theta_b) \tag{13}$$

と表すことができる．θ が空集合の時，U_a, U_b は U_p, U_q の共通鍵を作成可能である．また，U_a, U_b, U_p, U_q 以外のユーザを U_o とし，

$$\theta - \Theta_o = \phi \tag{14}$$

となるとき，U_o は U_p, U_q の共通鍵を作成可能である．

以降では，$\theta \neq \phi$ かつ $\theta - \Theta_o = \phi$ とし，θ に含まれるハッシュ値について考える．

h_j^i を計算できるユーザの集合を $G_{i,j}$ とすると，次のように表すことができる．

$$G_{i,j} := \{U_k \mid h_j^i \in \Theta_k\} \tag{15}$$

すると U_p, U_q が共通鍵を生成するのに必要な最少ハッ

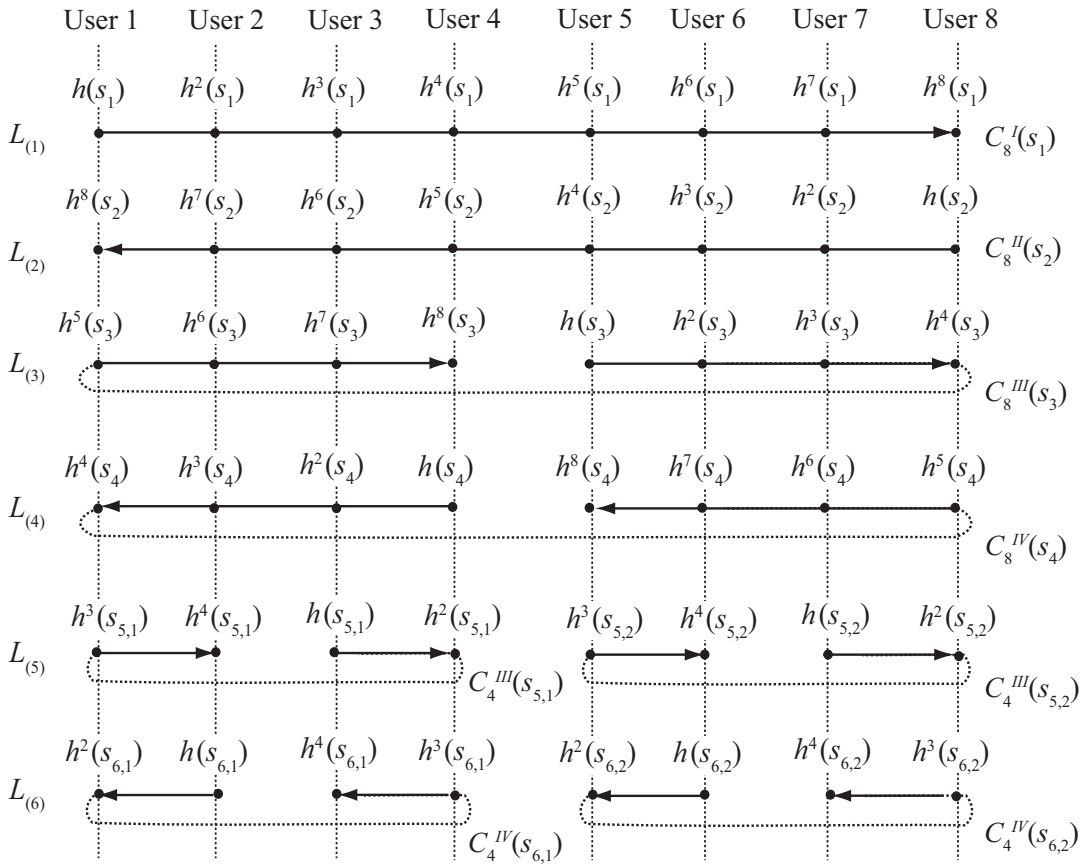

図 1 $N = 8$ のときの栗原の HCA

シュ値の集合は，全てのハッシュ値の集合を H とすると

$$B_{p,q} := \{T \subseteq H \mid \cap_{h_j^i \in T} G_{i,j} = \{U_p, U_q\}\} \tag{16}$$

となるハッシュ値の集合 $B_{p,q}$ が存在し，共通鍵生成に用いる最少のハッシュ値の集合は

$$E_{p,q} := \underset{T \in B_{p,q}}{\arg\min} \mid T \mid \tag{17}$$

と表すことができる．

4. おわりに

本稿では，栗原が提案したハッシュチェーンアグリゲーションと，それを改良したハッシュチェーンアグリゲーションの二つを，より一般的な方法で表現できることを示した．またハッシュチェーン計算モデルを用いて認証者が共通鍵を生成するのに必要な最少のハッシュ値の個数について述べた．

謝辞

本研究は科学研究費補助金 (課題番号 18K11300) 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] Y. Kurihara and M. Soshi, “A Novel Hash Chain Construction for Simple and Efficient Authentication,” Privacy, Security and Trust, Dec.2016.

[2] 平井晨太, 双紙正和. ハッシュチェーン計算モデルによる認証. 電子情報通信学会技術研究報告, Vol. 118, No. 486, ICSS2018-84, pp. 149-153, (2019)

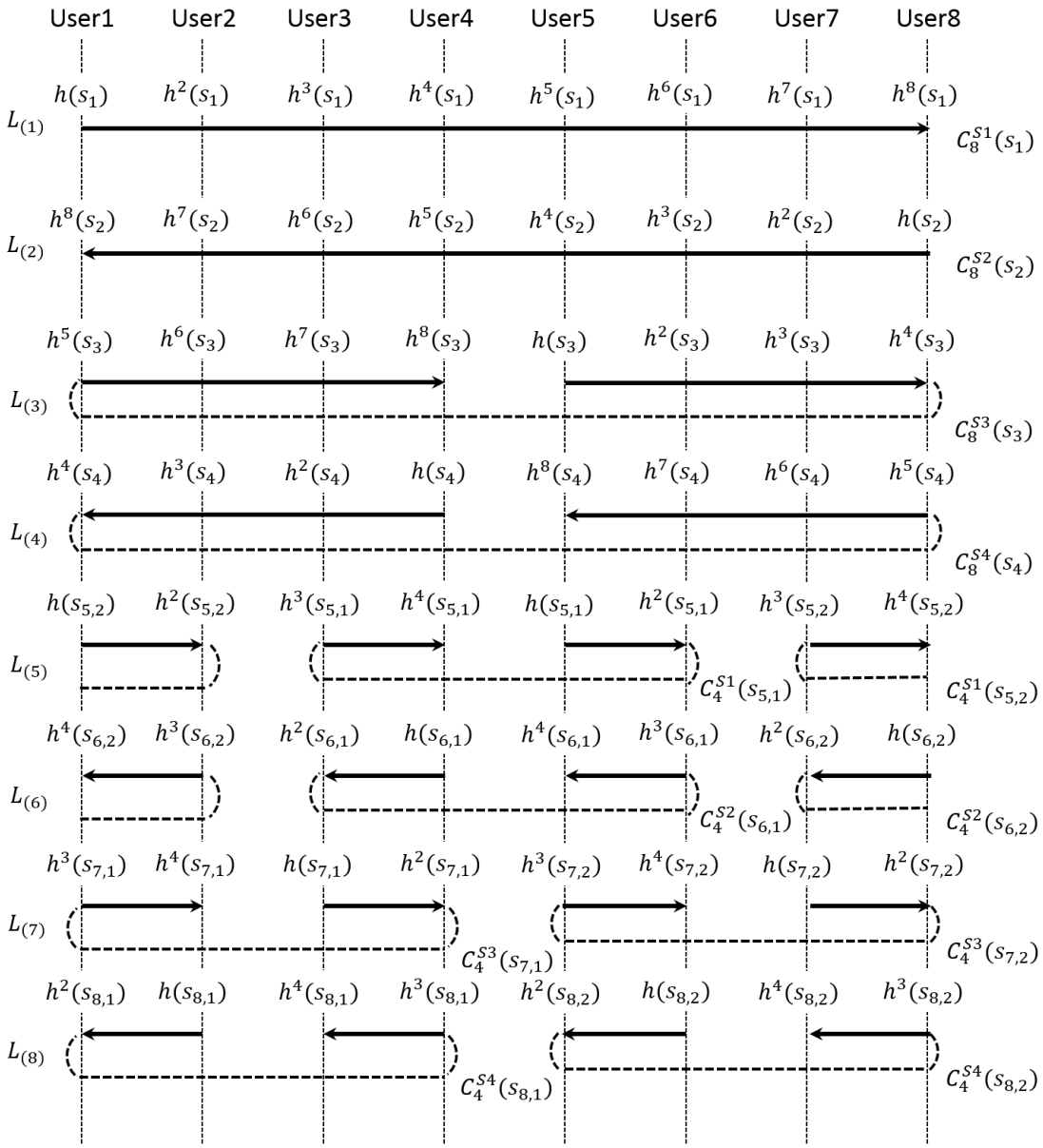

図 2 $N = 8$ のときの平井の HCA