効率の良い認証のための新しく柔軟なハッシュ連鎖構成法

栗原　勇太[1,a)]　双紙　正和[1,b)]

概要：近年，センサーネットワーク等において，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．本論文では，全く新しいハッシュ連鎖構成法 *hash chain aggregation* (HCA) を提案し，それを用いた，二者間での共通鍵生成法を提案する．本提案手法は，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こと，全く新しいハッシュ連鎖構成法に基づいていること，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ことを利点として持つ．

A Novel and Flexible Hash Chain Construction for Efficient Authentication

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

Abstract: In recent years, in sensor networks, etc.,the opportunity to connect with limited computing power and battery devices to the network is increasing and the importance of lightweight and efficient authentication protocols such as used hash function has been increasing. In this paper, we propose a completely novel hash chain construction *hash chain aggregation* (HCA) and a scheme to establish a common key for two users using it. Our proposed scheme has the following significant advantages, cryptographic primitives for our scheme are hash functions only and the resultant scheme is efficient, it is based on a totally new hash chain construction (HCA), what two users generating a common secret key must know is only the identity (ID) of the other. No communication is required except for in the initial setting. That is, our scheme is an ID-based authentication protocol.

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るなど，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．

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．また，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．一方で, 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]，現代の多くの暗号システムはそれらの困難性に基づいており，量子コンピュータを用いれば徐々に突破されていくであろう．

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

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

ない．

そこで，本論文では，最初に，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” を提案する．次に，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を提案する．本論文で提案するハッシュ連鎖構成法による認証は，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．

さらに本論文では，提案手法のセキュリティと性能を詳細に考察する．

本論文で提案するハッシュ連鎖アグリゲーションはこれまでに例を見ない全く新しいものである．そこで，実用上重要なだけでなく，ハッシュ連鎖構成法研究の新たな地平を切り開くものである．今後，ハッシュ連鎖アグリゲーションを応用した様々な軽量認証プロトコルの開発が期待される．

本論文は以下のように構成される．

2 節では，関連する研究について議論し，3 節では，新しいハッシュ連鎖構成法及び鍵生成手法を提案する．4 節では，提案した手法について，セキュリティ及び性能について詳細に評価を行う．最後に，5 にて結論を述べる．

2. 関連研究

この節では，ハッシュ連鎖を利用したセキュリティプロトコルについて，重要なものを選び，議論する．

Lamport は，ハッシュ連鎖を用いた認証プロトコルを提案した [7]．この手法はワンタイムパスワードと呼ばれ，パスワードは一回のみ用いられ，それがネットワーク上で漏洩しても問題ないという特徴がある．しかしながら，この手法は主としてユーザと一つのサーバのログインプロトコルとして用いられるもので，対等なユーザ同士の相互認証法としては適していない．

ハッシュ連鎖を用いた有名な認証プロトコルに，TESLA がある [8]．TESLA は，ハッシュ連鎖を用いた，効率の良い放送認証プロトコルであり，計算コストが低い，パケットロスに強いなどの重要な特徴を持つ． しかしながら，TESLA は，相互認証に用いるには複雑すぎ，またコストが高い．

Joye らは，ハッシュ連鎖を一般化し，one-way cross trees (OWCT) と呼ばれる手法を提案した．OWCT は，種の列の各要素に木のようにハッシュを適用しいてくことで，独立した秘密の値を作成する．OWCT は，key escrow などの応用を持つことが知られているが，ユーザ同士の共通の鍵を作成するのには適していない (少なくとも，OWCT によるそのような手法は知られていない)．

Dutta らは，ハッシュ連鎖を利用した，リボケーション可能な，self-healing key distribution を提案した [5]．その手法は，グループマネージャが，動的なグループのユーザにセッション鍵を配布し，パケットロスや共謀攻撃があったとしても，紛失された鍵を回復する手法である．この手法は，相互認証として用いるには複雑すぎ，またコストが大きい．一方で我々の手法は，いったん Key Genaration Center からハッシュ値が安全に分配されれば，以降は通信することなく，相手の ID のみで共通鍵を作成することができる．

3. 提案手法

本節では，新しいハッシュ連鎖構成法による，認証法 (二者間による共通鍵生成法) を提案する．

その前に，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．

3.1 基本的なアイディア

この節では，提案手法の基本的なアイディアについて述べる．

提案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 [*1]．

ここで，例として，$N = 5$ とし，ハッシュ連鎖を利用して，ユーザ $2, 4$ の共通鍵を作成する状況を考える．すると，もっとも単純には，KGC が，“2 方向ハッシュ連鎖” C_1, C_2 を用意する手法が考えられる．ここで，ある種 s_1, s_2 と，あるハッシュ関数 h について，$C_1 = (h(s_1), h^2(s_1), \ldots, h^5(s_1))$，$C_2 = (h^5(s_2), h^4(s_2), \ldots, h(s_2))$ である．そして，KGC は，これらのハッシュ値を計算した後，ユーザ i に，安全なチャンネルを介して $(h^i(s_1), h^{6-i}(s_2))$ を送る $(i = 1, 2, ..., 5)$．このとき，ユーザ i, j $(i < j$ とする) は，$h^j(s_1), h^{6-i}(s_2)$ の値を持っているか，あるいは，所有するハッシュ値から

[*1] 同等のモデル化として，各ユーザを，IoT あるいはセンサーネットワークにおけるノード (デバイス等) と考えることもできる．すなわち，KGC は秘密鍵を生成して各ノードに埋め込み，その後，ノードが deploy される．この場合は，安全なチャンネルは必要ない．

計算できることに注意せよ．すなわち，ユーザ i,j が認証のために共通鍵を作りたいときは，ある一方向性関数 F を用いて，$F(h^j(s_1), h^{6-i}(s_2))$ を共通鍵とすればよい．このとき，この共通鍵の作成には通信は必要なく，相手の ID のみで作成できる．

たとえば，ユーザ 2, 4 の場合について考えると，ユーザ 2, 4 はそれぞれ $(h^2(s_1), h^4(s_2))$, $(h^4(s_1), h^2(s_2))$ を配布されているので，

$$K_{2,4} := F(h^4(s_1), h^4(s_2)) \tag{2}$$

を共通鍵とすればよい．

3.2 基本的なアイディアの改良

3.1 節で考えた方式において，ハッシュ関数の一方向性より，ユーザ a は，$a < i$ または $j < a$ のとき，ユーザ i,j の共通鍵 $K_{i,j}$ を作ることはできない．たとえば，ユーザ 1, 5 は，$h^4(s_2), h^4(s_1)$ をそれぞれ作成することができないので，ユーザ 2, 4 の共通鍵 $K_{2,4}$ (Eq.(2)) を作成することができない．

しかしながら，$i < a < j$ であるユーザ a は，ユーザ i,j の共通鍵 $K_{i,j}$ を作成できる．たとえば，ユーザ 3 は，$(h^3(s_1), h^3(s_2))$ を持っているが，それから $h^4(s_1), h^4(s_2)$ を計算できるので，ユーザ 2, 4 の共通鍵 $K_{2,4}$ を作成できる．いうまでもなく，これは望ましくない．

そこで，鍵生成において，$i < a < j$ なるユーザ a を除外するようなハッシュ連鎖を考える．このために，本論文では，“非連続” なハッシュ連鎖 C_3, C_4 を考える．C_3, C_4 の形式的な定義は以降の節で与えるが，ここではそれを直感的に理解するために，3.1 節と同様，$N = 5$ の場合について考えよう．

ある種 s_3, s_4 について，KGC は，$h(s_3), ..., h^5(s_3)$ および $h(s_4), ..., h^5(s_4)$ をそれぞれ順に計算し，それらを並び替えて，以下のようなハッシュ連鎖 C_3, C_4 を構成する：

$$C_3 = (h^3(s_3), h^4(s_3), h^5(s_3), h(s_3), h^2(s_3)) \tag{3}$$
$$C_4 = (h^3(s_4), h^2(s_4), h(s_4), h^5(s_4), h^4(s_4)) \tag{4}$$

このとき，C_3, C_4 の各ハッシュ値は，

$$h^{((i-\lceil N/2\rceil-1) \bmod N)+1}(s_3), \text{and}$$
$$h^{((\lceil N/2\rceil-i) \bmod N)+1}(s_4)$$

とそれぞれ書くことができる $(1 \le i \le N)$.

Eq.(3) を見れば分かるように，C_3 は，ハッシュ関数のインデントが単純に増加していく従来のハッシュ連鎖 (たとえば C_1 など) を二つの部分に分け，それを交換したものである (C_2, C_4 の関係も，同様に考えることができる)．このようなハッシュ連鎖の構成はこれまでにない新しいものであり，このようにハッシュ連鎖を柔軟に構成することで，単純で効率のよい鍵生成が可能となる．

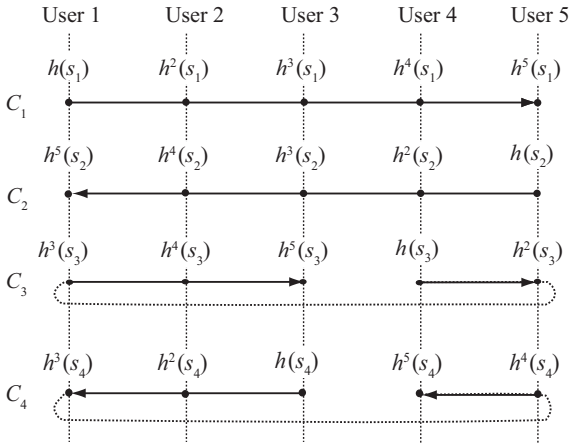

図 1 ハッシュ連鎖 C_1, C_2, C_3, C_4
Fig. 1 Hash Chains C_1, C_2, C_3, C_4

まとめると，認証法は以下のようになる．KGC は，C_1, C_2, C_3, C_4 を構成し，ユーザ i $(1 \le i \le N)$ に，以下のような 4 個のハッシュ値を安全なチャンネルを用いて安全に配送する (図 1 参照)：

$$h^i(s_1), h^{N-i+1}(s_2), h^{((i-\lceil N/2\rceil-1) \bmod N)+1}(s_3),$$
$$h^{((\lceil N/2\rceil-i) \bmod N)+1}(s_4)$$

そして，ユーザ i,j は，認証のため共通鍵を作成したいとき，

$$h^j(s_1), h^{N-i+1}(s_2), h^{((i-\lceil N/2\rceil-1) \bmod N)+1}(s_3),$$
$$h^{((\lceil N/2\rceil-j) \bmod N)+1}(s_4) \tag{5}$$

を計算し，これら 4 個の値を一方向性関数 F の入力とすることで，共通鍵を作成する．

再び $N = 5$ の場合を考える．このとき，ユーザ 2, 3, 4 は，それぞれ $(h^2(s_1), h^4(s_2), h^4(s_3), h^2(s_4))$, $(h^3(s_1), h^3(s_2), h^5(s_3), h(s_4))$, $(h^4(s_1), h^2(s_2), h(s_3), h^5(s_4))$ を与えられる．そして，ユーザ 2, 4 は，共通鍵

$$F(h^4(s_1), h^4(s_2), h^4(s_3), h^5(s_4)) \tag{6}$$

を作成することができる．このとき，ユーザ 3 は，ハッシュ関数 h の一方向性より，$h^4(s_3)$ を計算できないので，式 (6) を計算することができない．

同様にして，$N = 5$ のとき，ユーザ 1, 3 の組み合わせを除くと，任意の i,j $(1 \le i < j \le 5)$ について，ユーザ i,j 以外のユーザは，ユーザ i,j の共通鍵を作成できないことも容易に確かめられる．

3.3 新しいハッシュ連鎖構成法

3.2 節で述べた手法では，ハッシュ連鎖は 4 個しかなく，ユーザ 1, 3 の組み合わせや，N が大きい場合などに対処できない．そこでこの節では，より一般的な，新たなハッ

シュ連鎖の構成法を提案する．

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．

3.3.1 基本的なハッシュ連鎖

提案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

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

- タイプ II ハッシュ連鎖

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

- タイプ III ハッシュ連鎖

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

- タイプ IV ハッシュ連鎖

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

3.3.2 ハッシュ連鎖リスト

3.3.1 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．

$$\begin{aligned}&\mathcal{L}(\tau, k, s_1, ..., s_k)\\ &:= (C^\tau_{N/k}(s_1), C^\tau_{N/k}(s_2), ..., C^\tau_{N/k}(s_k))\end{aligned} \tag{7}$$

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．

3.3.3 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．

$$\mathcal{A} := \{\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_r\} \tag{8}$$

$\mathcal{L}_i$ $(1 \leq i \leq r)$ は，式 (7) によって定義される．

3.3.4 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 (8) 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，

$$\frac{N}{k_\mathcal{L}}(d-1) + 1 \leq i \leq \frac{N}{k_\mathcal{L}} d \quad (1 \leq d \leq k_\mathcal{L}) \ . \tag{9}$$

式 (9) 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 [*2]．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 [*3]．

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，

$$V(\mathcal{A}, i) := \left\{ (\mathcal{L}[\alpha(\mathcal{L}, i)]) \left[i - \frac{N}{k_\mathcal{L}} \cdot (\alpha(\mathcal{L}, i) - 1) \right] \,\middle|\, \mathcal{L} \in \mathcal{A} \right\} \ . \tag{10}$$

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，$\mathcal{A}$ においてノード i を含むハッシュ連鎖の集合は，

$$W(\mathcal{A}, i) := \{\mathcal{L}[\alpha(\mathcal{L}, i)] \mid \mathcal{L} \in \mathcal{A}\} \tag{11}$$

と定義される関数 W によって表現できる．

3.4 提案ハッシュ連鎖アグリゲーション

提案する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．

$$\mathcal{HCA} := (\mathcal{L}_{(1)}, \mathcal{L}_{(2)}, \ldots, \mathcal{L}_{(2m-1)}, \mathcal{L}_{(2m)}) \tag{12}$$

ここで，$\mathcal{L}_{(1)} = (C^I_N(s_1))$, $\mathcal{L}_{(2)} = (C^{II}_N(s_2))$, $\mathcal{L}_{(3)} = (C^{III}_N(s_3))$, $\mathcal{L}_{(4)} = (C^{IV}_N(s_4))$ 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，$3 \leq j \leq m$ である．

[*2] すなわち，$\alpha(\mathcal{L}, i) = \lceil ki/N \rceil$ であり，$\alpha(\mathcal{L}, i)$ 番目のハッシュ連鎖 $L \in \mathcal{L}$ が，(ハッシュ連鎖リスト $\mathcal{L}$ のすべてのハッシュ値に関して) i 番目のハッシュ値を含むということを意味する．

[*3] $Q[j]$ は，列 Q の j 番目の要素であることに注意．

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 2 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，$W(\mathcal{HCA}, 3) = \{C^I_8(s_1), C^{II}_8(s_2), C^{III}_8(s_3), C^{IV}_8(s_4), C^{III}_4(s_{5,1}), C^{IV}_4(s_{6,1})\}$ である．

3.5 $\mathcal{HCA}$ 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，

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

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:

$$\begin{aligned} &\nu_1 := C^I_N(s_1)[j], \\ &\nu_2 := C^{II}_N(s_2)[i], \\ &\nu_3 := (\mathcal{L}_{(2k_u-1)}[\alpha(\mathcal{L}_{(2k_u-1)}, i)])[i - \beta_{i,2k_u-1}], \\ &\nu_4 := (\mathcal{L}_{(2k_u)}[\alpha(\mathcal{L}_{(2k_u)}, i)])[j - \beta_{j,2k_u}] \end{aligned} \tag{13}$$

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，

$$K_{i,j} := F(\nu_1, \nu_2, \nu_3, \nu_4)\ . \tag{14}$$

と計算すればよい．

3.6 例

例として，$N = 8$ の場合の $\mathcal{HCA}$ (図 2 参照) を考える．

- $i = 3, j = 6$ のとき
  $W(\mathcal{HCA}, 3) = \{C^I_8(s_1), C^{II}_8(s_2), C^{III}_8(s_3), C^{IV}_8(s_4), C^{III}_4(s_{5,1}), C^{IV}_4(s_{6,1})\}$, $W(\mathcal{HCA}, 6) = \{C^I_8(s_1), C^{II}_8(s_2), C^{III}_8(s_3), C^{IV}_8(s_4), C^{III}_4(s_{5,2}), C^{IV}_4(s_{6,2})\}$. よって，$W(\mathcal{HCA}, 3) \cap W(\mathcal{HCA}, 6) = \{C^I_8(s_1), C^{II}_8(s_2), C^{III}_8(s_3), C^{IV}_8(s_4)\}$．これより，$k_1 = k_u = 2$ であり，$\alpha(\mathcal{L}_{(3)}, 3) = \alpha(\mathcal{L}_{(3)}, 6) = \alpha(\mathcal{L}_{(4)}, 3) = \alpha(\mathcal{L}_{(4)}, 6) = 1$, $\beta_{3,3} = \beta_{6,4} = 0$ となるので，

  $$\begin{aligned} \nu_1 &= C^I_8(s_1)[6] = h^6(s_1), \\ \nu_2 &= C^{II}_8(s_2)[3] = h^6(s_2), \\ \nu_3 &= (\mathcal{L}_{(3)}[1])[3] = C^{III}_8(s_3)[3] = h^7(s_3), \\ \nu_4 &= (\mathcal{L}_{(4)}[1])[6] = C^{IV}_8(s_4)[6] = h^7(s_4) \end{aligned}$$

  と計算でき，$K_{i,j}$ を得ることができる．
- $i = 5, j = 7$ のとき
  このとき，$W(\mathcal{HCA}, 5) \cap W(\mathcal{HCA}, 7) = \{C^I_8(s_1), C^{II}_8(s_2), C^{III}_4(s_{5,2}), C^{IV}_4(s_{6,2})\}$．これより，

  $$\begin{aligned} \nu_1 &= C^I_8(s_1)[7] = h^7(s_1), \\ \nu_2 &= C^{II}_8(s_2)[5] = h^4(s_2), \\ \nu_3 &= (\mathcal{L}_{(5)}[2])[5-4] = C^{III}_4(s_{5,2})[1] = h^3(s_{5,2}), \\ \nu_4 &= (\mathcal{L}_{(6)}[2])[7-4] = C^{IV}_4(s_{6,2})[3] = h^4(s_{6,2}) \end{aligned}$$

  と計算できる．

いずれの場合においても，ユーザ i, j は，自らの情報のみで共通鍵 $K_{i,j}$ を計算でき，かつ，ユーザ i, j 以外のユーザは，$K_{i,j}$ を作成できないことを，容易に確認できる．

提案手法のセキュリティについては，4.1 節で詳しく検討する．

4. 評価

ここでは，提案手法について，セキュリティと性能の評価を行う．

4.1 セキュリティ評価

提案方式は，ハッシュ連鎖を利用した，相互認証のための鍵生成法を実現するものである．KGC が，secure channel を介して各ユーザにハッシュ値を配送し，またハッシュ関数の性質から，ユーザ 1, ..., N 以外の攻撃者は，ハッシュ値を計算することができない．そこで，提案手法に対する攻撃は，攻撃者が何らかの手段であるユーザのハッシュ値を入手するか，あるいは，ユーザ自身が攻撃者であり，自分のハッシュ値を利用して，他のユーザ i, j の共通鍵を計算するような状況しかない．いずれにせよ，あるユーザが攻撃者である状況を考えればよい．

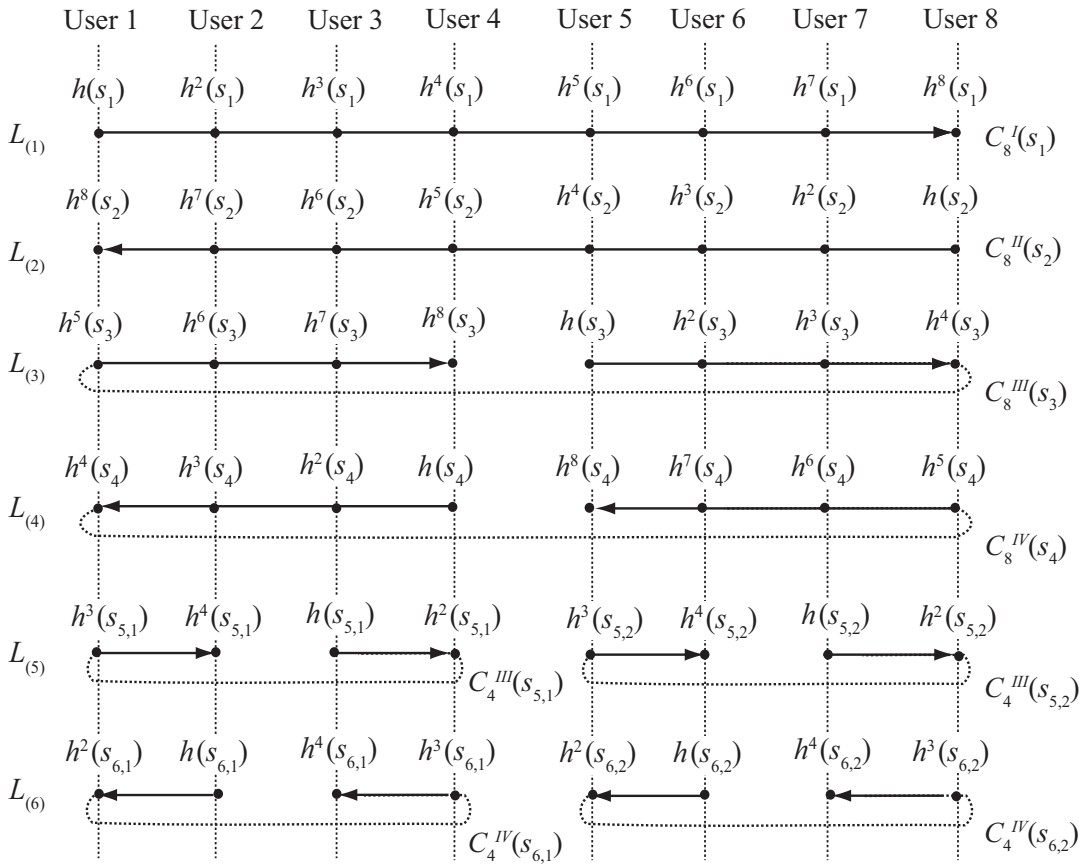

図 2 $N = 8$ のときの $\mathcal{HCA}$
Fig. 2 Our $\mathcal{HCA}$ ($N = 8$)

4.1.1 攻撃者が一人である場合

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．

Proof. 上の議論より，ユーザ a $(1 \le a \le N)$ が攻撃者である場合だけ考えればよい．$K_{i,j}$ を計算するには，式 (13) より，少なくとも $C_N^I(s_1)[j], C_N^{II}(s_2)[i]$ が必要である．しかしながら，$a < i$ の場合は，ハッシュ関数 h の一方向性より，ユーザ (攻撃者) a は，$C_N^{II}(s_2)[i]$ を計算できない．同様に，$j < a$ の場合は，ユーザ a は，$C_N^I(s_1)[j]$ を計算できない．よって，$a < i$ あるいは $j < a$ の場合は，ユーザ a は $K_{i,j}$ を計算できない．

ここで，$i < a < j$ なるユーザ a は，$C_N^I(s_1)[j], C_N^{II}(s_2)[i]$ を計算することができる．このとき，$i+1 \neq j$ である．しかしながら，このユーザ a は，式 (13) における ν_3 あるいは ν_4 を計算することができない．これを以下で証明する．

まず，式 (13) より，$\mathcal{L}_{(2k_u-1)}, \mathcal{L}_{(2k_u)}$ について考えると，それらは，

$$\mathcal{L}_{(2k_u-1)} = \mathcal{L}(III, 2^{k_u-2}, s_{2k_u-1,1}, ..., s_{2k_u-1,2^{k_u-2}})$$
$$\mathcal{L}_{(2k_u)} = \mathcal{L}(IV, 2^{k_u-2}, s_{2k_u,1}, ..., s_{2k_u,2^{k_u-2}})$$

と書けることに注意せよ．ここで，$k := 2^{k_u-2}$ とおく．また，3.5 節で述べたように，$\alpha(\mathcal{L}_{(2k_u-1)}, i) = \alpha(\mathcal{L}_{(2k_u-1)}, j) = \alpha(\mathcal{L}_{(2k_u)}, i) = \alpha(\mathcal{L}_{(2k_u)}, j)$ なので，この値を単に α と書く．

$i < a < j$ を以下の二つの場合に分けて証明する．このとき，k_u の選び方および $i+1 \neq j$ より，$i \le N(\alpha-1)/k+N/2k$ かつ $N(\alpha-1)/k + N/2k + 1 \le j$ となる [*4]．

1. $i < a \le N(\alpha-1)/k+N/2k$ のとき．このときユーザ a は，ν_3 を計算することができないため，$K_{i,j}$ を計算できない．
2. $N(\alpha-1)/k+N/2k+1 \le a < j$ のとき．このときユーザ a は，ν_4 を計算することができない．

以上より，定理が証明された． □

4.1.2 結託攻撃

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 $K_{i,j}$

[*4] もし $i < j \le N(\alpha-1)/k+N/2k$ あるいは $N(\alpha-1)/k+N/2k+1 \le i < j$ となるとすれば，$k_u < k'$ なる k' で，$\mathcal{L}_{(2k'-1)}[\alpha(\mathcal{L}_{(2k'-1)}, i)]$, $\mathcal{L}_{(2k')}[\alpha(\mathcal{L}_{(2k')}, i)] \in W(\mathcal{HCA}, i) \cap W(\mathcal{HCA}, j)$ となるものが存在するからである．これは前提に反する．

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．

Proof. まず，式 (13) を参照せよ．ユーザ a_1, a_2 は，いずれも ν_1, ν_2 を計算できる．また，ユーザ a_1 は，ν_4 を計算できる (ν_3 は計算できない)．ユーザ a_2 は，ν_3 を計算できる (ν_4 は計算できない)．以上より，ユーザ a_1, a_2 が結託することで，$K_{i,j}$ を計算できる． □

以上より，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．本提案方式は，比較的信頼される環境で使用されるべきである．

4.2 性能

この節では，提案手法の性能を評価する．

4.2.1 各ユーザが保持するハッシュ値の数

ユーザの数を N とすると，本提案手法では，各ユーザは $2\lceil \log_2 N \rceil$ 個のハッシュ値を保持しなければならない．共通鍵によって相互認証する自明な手法だと，各ユーザは $N-1$ 個の秘密鍵を持たなければならないので，本提案手法は非常に効率が良い．

4.2.2 計算量

ユーザ i, j $(1 \leq i < j \leq N)$ が共通鍵 $K_{i,j}$ を計算するために必要なハッシュ回数を評価する．式 (13) を参照せよ．ユーザ i は，ν_2, ν_3 を所持しているので，計算は必要ない．ν_1 を得るために，ハッシュ関数 h の計算が $j-i$ 回必要である．また，ν_4 を得るために，$(i-j) \bmod (N/k)$ 回 h を計算しなければならない．ここで，$k = 2^{k_u - 2}$ とおいた．以上より，ユーザ i が共通鍵を計算するためには，F の計算を除くと，ハッシュ関数 h を

$$j - i + (i-j) \bmod \frac{N}{k}$$

回計算する必要がある．この値は $O(N)$ であるから，N が大きいときは，共通鍵作成のための計算量が大きくなることが言える．

しかしながら，近年のハッシュ関数，たとえば SHA-2 や SHA-3 は非常に高速な実装が可能であり [11]，実用上は大きな問題ではないと考えられる．

4.2.3 共通鍵計算に必要なハッシュ値の個数

ユーザ i, j $(1 \leq i < j \leq N)$ の共通鍵 $K_{i,j}$ を他のユーザが作成できないように，式 (13) では 4 個のハッシュ値 $(\nu_1, \nu_2, \nu_3, \nu_4)$ を必要とした．しかしながら，i, j の状況によっては，より少ない個数のハッシュ値で，$K_{i,j}$ を作ることができる [*5]．

以下では，α, k を，定理 1 と同様に定義する．また，以下の例では，$N = 8$ とする (図 2 参照)．

(1) $i + 1 = j$ のとき．
2 個のハッシュ値 ν_1, ν_2 で十分である．
また，タイプ III, IV のハッシュ連鎖においては，ユーザ $N(\alpha-1)k+1$ とユーザ $N\alpha/k$ は隣接しているので，2 個のハッシュ値 ν_3, ν_4 で十分である．
例: ユーザ 1, 8 のとき，$\nu_3 = C_8^{III}(s_3)[1] = h^5(s_3)$, $\nu_4 = C_8^{IV}(s_3)[8] = h^5(s_4)$.

(2) $i = N(\alpha-1)/k + N/2k$, $j = N\alpha/k$ のとき
2 個のハッシュ値 ν_2, ν_4 で十分である．
例: ユーザ 4, 8 のとき，$\nu_2 = C_8^{II}(s_2)[4] = h^5(s_2)$, $\nu_4 = C_8^{IV}(s_4)[8] = h^5(s_4)$.

(3) $i = N(\alpha-1)/k + 1$, $j = N(\alpha-1)/k + N/2k + 1$ のとき
2 個のハッシュ値 ν_1, ν_3 で十分である．
例: ユーザ 5, 7 のとき，$\nu_1 = C_8^{I}(s_1)[7] = h^7(s_1)$, $\nu_3 = C_4^{III}(s_{5,2})[1] = h^3(s_{5,2})$.

(4) $i = N(\alpha-1)/k + N/2k$, $N(\alpha-1)/k + N/2k + 1 < j < N\alpha/k$ のとき
3 個のハッシュ値 ν_1, ν_2, ν_4 で十分である．
例: ユーザ 4, 6 のとき，$\nu_1 = C_8^{I}(s_1)[6] = h^6(s_1)$, $\nu_2 = C_8^{II}(s_2)[4] = h^5(s_2)$, $\nu_4 = C_8^{IV}(s_4)[6] = h^7(s_4)$.

(5) $1 < i < N(\alpha-1)/k + N/2k$, $j = N(\alpha-1)/k + N/2k + 1$ のとき．
3 個のハッシュ値 ν_1, ν_2, ν_3 で十分である．
例: ユーザ 3, 5 のとき，$\nu_1 = C_8^{I}(s_1)[5] = h^5(s_1)$, $\nu_2 = C_8^{II}(s_2)[3] = h^6(s_2)$, $\nu_3 = C_8^{III}(s_3)[3] = h^7(s_3)$.

[*5] 紙面の制約により，証明は省略し，事実のみ列挙する．

5. 結論

本論文では，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” を提案した．次に，それを用いた，二者間での共通鍵生成法を提案した．本提案手法は，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．

さらに，本論文では，提案手法のセキュリティと性能を詳細に考察した．

本論文で提案したハッシュ連鎖アグリゲーションは，これまでに例を見ない全く新しいもので，ハッシュ連鎖構成法研究の新たな地平を切り開くものである．この構成法のさらなる応用を検討することが，今後の課題である．

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.