Vol.2021-DPS-188 No.25
Vol.2021-EIP-93 No.25
2021/9/10

非対称な鍵共有アルゴリズムについての考察と実装

神保　洸貴[1,a)]　入山　聖史[2,b)]

概要：2011 年，Accardi らにより考案された Strongly Asymmetric Public Key Agreement(SAPKA) と呼ばれる非対称な公開鍵共有フレームワークでは，Alice と Bob 間で鍵共有を行う際，従来の「対称」な鍵共有アルゴリズムとは異なり，公開鍵数，公開鍵・共有鍵 (SSK) 計算規則が Alice と Bob で対称的でない．Bob の公開鍵数，公開鍵・SSK の計算量は Alice よりも多く，攻撃者 (Eve) は従来とは異なるプロセスにより SSK の導出を試みる必要があり，Bob の公開鍵のセキュリティ強度が十分高ければ，Alice の公開鍵のセキュリティ強度は Bob のそれほど重要ではなくなる．その結果 Alice は自身の公開鍵の計算量を落とすことができる場合がある．本研究では，Diffie-Hellman 鍵共有アルゴリズムを SAPKA フレームワークの規則に準じて書き直し，非可換代数上に拡大することで，Alice と Bob で計算量が非対称な鍵共有アルゴリズムの構築を行う．また，簡易的な実装実験と従来の Diffie-Hellman との性能比較を行うことで，Alice の計算コストがどれだけ低下したかを評価する．

キーワード：IoT，鍵共有，非対称，Diffie-Hellman 鍵共有アルゴリズム，耐量子

1. はじめに

安全でない通信上で，Alice と Bob 間で安全に暗号文のやりとりを行う RSA 公開鍵暗号 [1] や安全に秘密鍵を交換する Diffie-Hellman(D-H) 鍵共有アルゴリズム [2] の考案から約 50 年が経った．安全な暗号通信システムの構築のために，RSA 暗号や D-H アルゴリズムは今なお用いられており，データ保護や個人情報漏洩の観点で大きな役割を担っている．

しかし，近年の計算機科学の著しい発展により，これらのアルゴリズムの安全性が危惧されている．計算機科学の発展による恩恵を享受できるのは一般的なユーザーだけではない．悪意のあるユーザー (以後 Eve と表記) が攻撃に用いることができる計算機の性能も過去と比較して格段に向上しており，短いビット数の共有鍵 (e.g, 512bit) を用いた暗号通信はもはや安全とはいえない状況にある [3]．また，安全基準とされる鍵長も年々増加しており，鍵長の増大は計算コストの増大に大きな関連がある．IoT デバイスの普及により計算能力の限られたデバイス上で暗号通信を行う場面は増加すると考えられ，これらのデバイスを用いた安全な暗号通信が不可能になる恐れがある．実際，安全基準を下回る鍵長の D-H や RSA で鍵共有を行うユーザも増えており [4]，RSA や D-H の代わりとなる高速化かつ安全な鍵共有・公開鍵暗号アルゴリズムの研究・開発が急務である．

また，1997 年，因数分解問題や離散対数問題が量子計算機を用いたアルゴリズムにより多項式時間内に解読される可能性が Shor により指摘されており [5]，近年研究開発が盛んな量子計算機とそれを用いた量子アルゴリズムによる脅威も考慮しなければならない．

これらの問題に対して，耐量子暗号 (PQC) と呼ばれる量子計算機を用いたとしても解読が難しい暗号が現在盛んに研究されており，代表的なものとして最近・最短ベクトル問題や Learn-with-Error 問題など格子問題 [6], [7], [8] の困難性をもとにした NTRU[9] や New-Hope[10] などがある．しかし鍵サイズが非常に大きいことや特定のパラメータ (素数のビット数，行列の次元，非直行基底など) を使用すると鍵長が大きくても安全性が低下する場合がある (NTRU 方式における脆弱パラメータ [11]，LWE 方式による脆弱パラメータ [12]) など，これらを D-H や RSA の代わりとして使用したり，計算能力の限られたデバイス上で使用するためには不明瞭な点が未だ多いことが指摘されており，実用化にはまだ時間を要する．

あらゆるデバイス上で安全な通信を行うために，PQC など安全かつ高速な鍵共有・公開鍵暗号アルゴリズムの考案を目指す研究は現在の暗号理論分野で活発に行われている

[1] 東京理科大学
理工学研究科 情報科学専攻〒 278-8510　千葉県野田市山崎 2641
[2] 東京理科大学
理工学部 情報科学科〒 278-8510　千葉県野田市山崎 2641
[a)] 6319702@ed.tus.ac.jp
[b)] iriyama@is.noda.tus.ac.jp

アプローチの一つであるが，我々は以下に述べるような方法も有効なアプローチの一つではないかと考える．

1.1 研究概念と目的

計算能力の限られたデバイス上で安全な暗号通信が困難になり得るという問題は，以下からも説明できる．前鍵共有アルゴリズムの集合を PKA とおき，ある鍵共有アルゴリズム $alg \in PKA$ における共有秘密鍵 (SSK) のビット数に対する Alice，Bob が SSK 計算に必要な計算ステップ数をそれぞれ以下の N に対する単調増加関数，

$$T_A : PKA \times \mathbb{N} \to \mathbb{N} \tag{1}$$

$$T_B : PKA \times \mathbb{N} \to \mathbb{N} \tag{2}$$

で表す．D-H など従来のアルゴリズムにおいては一般的に以下，

$$T_A(alg, N) = T_B(alg, N) \tag{3}$$

が成立している．この等式が成立するアルゴリズム $alg_{A=B} \in PKA$ に対して，Alice のデバイスの計算能力 $E_A \in \mathbb{R}^+$(単位時間当たりの計算ステップ数) が Bob のデバイスのそれ E_B よりも低い ($E_A < E_B$) と想定する．任意の $alg \in PKA$ における N ビットの SSK を計算するのに Alice と Bob が必要な計算コスト (時間) をそれぞれ以下

$$C_A(T_A(alg, N), E_A) := \frac{T_A(alg, N)}{E_A} \tag{4}$$

$$C_B(T_B(alg, N), E_B) := \frac{T_B(alg, N)}{E_B} \tag{5}$$

で表すと，$T_A(alg_{A=B}, N) = T_B(alg_{A=B}, N), 1/E_A > 1/E_B$ より，

$$\begin{aligned} C_A(T_A(alg_{A=B}, N), E_A) &= \frac{T_A(alg_{A=B}, N)}{E_A} \\ &> C_B(T_B(alg_{A=B}, N), E_B) \\ &= = \frac{T_B(alg_{A=B}, N)}{E_B} \end{aligned} \tag{6}$$

が任意の N で成立する．このとき，ある N_0 を安全性を保つのに必要な最小の SSK ビット長とし，Bob の計算時間を基準として両者は $C_{max} := C_B(T_B(alg_{A=B}, N_0), E_B) = \frac{T_B(alg_{A=B}, N_0)}{E_B}$ 時間内で鍵共有を行うと仮定する．この際 Alice が計算可能な SSK ビット長 N は，以下により制限される．

$$C_A(T_A(alg_{A=B}, N), E_A) \leq C_{max} \tag{7}$$

$\frac{1}{E_A} > \frac{1}{E_B}$ より Alice は C_{max} 時間以内で，ある N_1 ビット ($N_1 < N_0$) までの SSK しか計算できない．結局，両者は N_1 ビットの SSK で鍵共有を行うか，C_{max} より大きな時間で鍵共有を行うしかない．IoT 化が進む現代社会において片方のデバイスの計算能力が低い場面は多く存在し，鍵共有の安全性はビット数に依存すること，IoT 機器における計算コストの増大は利便性を阻害することを考えると，どちらの選択肢も避けたいところである．

一方で，もし全ての N において $T_A(alg_{A<B}, N) < T_B(alg_{A<B}, N)$ を満たすアルゴリズム $alg_{A<B}$ が存在することを仮定してみる．このとき $alg_{A=B}$ の場合と同様に両者は，Bob の計算時間を基準として $C_{max} := C_B(T_B(alg_{A=B}, N_0), E_B) = \frac{T_B(alg_{A=B}, N_0)}{E_B}$ 時間内で鍵共有を行うとすると Alice がこの時間内に計算可能な SSK ビット長は

$$\begin{aligned} C_A(T_A(alg_{A<B}, N), E_A) &= \frac{T_A(alg_{A<B}, N)}{E_A} \\ &\leq C_{max} = \frac{T_B(alg_{A<B}, N_0)}{E_B} \end{aligned} \tag{8}$$

により与えられる．したがって以下の条件

$$\frac{T_A(alg_{A<B}, N_0)}{T_B(alg_{A<B}, N_0)} \leq \frac{E_A}{E_B} \tag{9}$$

が成立していれば，Alice は C_{max} 時間以内に N_0 ビットの SSK を計算することができる．もちろん，与えられた $\frac{E_A}{E_B}$ によっては (9) が成立するとは限らないことに注意しなくてはならないが，$alg_{A=B}$ の場合には (9) が成立しないことを考慮すると計算量が Alice と Bob 間で異なるアルゴリズム $alg_{A<B}$ の発見は，計算能力の限られたデバイスを用いた安全な暗号通信の実現に寄与できる可能性がある．本論文では，2011 年に Accardi らにより考案された，Strongly Asymmetric Public Key Agreement(SAPKA)[13] と呼ばれる非対称な鍵共有フレームワークを用いて，計算量が Alice と Bob 間で異なる鍵共有アルゴリズムの考案とその実装実験を行う．

2. 研究手法

2.1 SAPKA

ここでは [13] にて考案された非対称な鍵共有フレームワークを簡単に説明する．

SAPKA は任意の乗法半群 $\mathcal{S}$ 上における以下の 5 つの写像

$$x_1, x_2, x_3, x_4, N_1 : \mathcal{S} \to \mathcal{S} \tag{10}$$

を用いて各鍵共有プロセスが記述されており，各写像を具体的に定めることで実装が可能な鍵共有アルゴリズムを記述することができるフレームワークである．以下の条件

- 写像 N が可逆であり，N から逆写像 N^{-1} の導出および写像の合成 $N^{-1} \circ N$ が容易 (多項式時間内) に行えること
- 全ての $y \in \mathcal{S}$ に対して

$$x_1 \circ x_2(y) = x_3 \circ x_4(y) \tag{11}$$

が成立していれば，図 1 のように各鍵共有プロセスを記述することができる．

「非対称」とは，Bob の公開鍵数が 2 つなのに対して，Alice の公開鍵数は 1 と異なるところにあり，結果として

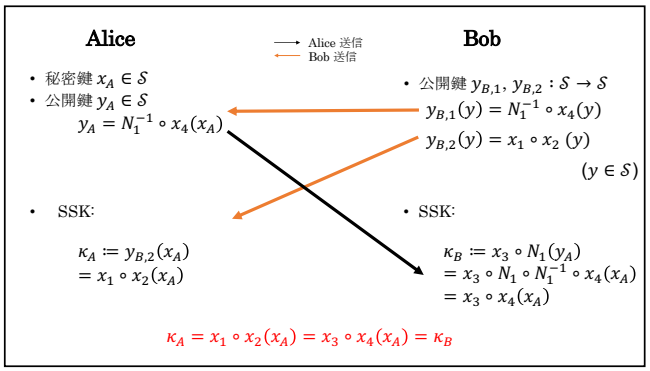

図 1　SAPKA の各鍵共有プロセス
Fig. 1　Key Agreement Process of SAPKA

Alice と Bob はそれぞれ異なる計算規則により SSK を計算することになる．この特性により攻撃者 (Eve) は Alice もしくは Bob の秘密鍵を得るために最大 2 回，公開鍵に対して攻撃を試みなくてはならず，その際 Alice は自身の秘密鍵の空間を小さくしても安全性を損なわない可能性がある．以下で SAPKA クラスに対する一般的な攻撃方法を考察し，研究手法を具体的に定める．

2.2　SAPKA に対する攻撃

[14] にて Jimbo, Iriyama, Regoli により考察されている，SAPKA に対する Eve の攻撃手法を以下に述べる．ただしここでは中間者攻撃については考慮しないこととする．Eve は図 1 における κ_A もしくは κ_B を以下の公開鍵

$$y_{B,1}(y) = N_1^{-1} \circ x_4(y) \qquad (y \in \mathcal{S}) \tag{12}$$

$$y_{B,2}(y) = x_1 \circ x_2(y) \qquad (y \in \mathcal{S}) \tag{13}$$

$$y_A = N_1^{-1} \circ x_4(x_A) \tag{14}$$

から導出することを目指す．ここで $y_{B,1}$，$y_{B,2}$ は $\mathcal{S}$ 上の写像であり，y_A は $\mathcal{S}$ の要素である．

2.2.1　パターン 1

もし x_4 が可逆であれば $y_{B,1}$ も可逆である．この時 Eve は，(12)，(14) を用いて Alice の秘密鍵 x_A の導出を以下のように

$$\begin{aligned} x_A &= y_{B,1}^{-1}(y_A) \\ &= x_4^{-1} \circ N_1 \circ N_1^{-1} \circ x_4(x_A) \end{aligned} \tag{15}$$

のように試みる．Eve は x_A を用いて SSK

$$\kappa_A = x_1 \circ x_2(x_A) \tag{16}$$

を計算できる．

もし x_4 が可逆でなければ，Eve は上記 x_A の代わりに現像 $\{x_E \in \mathcal{S} : y_A = N_1^{-1} \circ x_4(x_E)\}$ を得る．この時，この現像から任意に要素 x_E を取り出し以下の計算に用いることで

$$\begin{aligned} x_1 \circ x_2(x_E) &= x_3 \circ x_4(x_E) \\ &= x_3 \circ N_1 \circ N_1^{-1} \circ x_4(x_E) \\ &= x_3 \circ N_1(y_A) = \kappa_B \end{aligned} \tag{17}$$

x_4 が可逆である場合と同様に SSK を計算することができる．ここで 2 つめの等式は (11) より，最後の等式は κ_B の定義より導き出せる．

2.2.2　パターン 2

Eve は (12) から N_1 の導出を試みる．その後以下が全ての $y \in \mathcal{S}$ で成り立つような写像 $x_{3,E} : \mathcal{S} \to \mathcal{S}$

$$x_{3,E} \circ N_1 \circ N_1^{-1} \circ x_4(y) = x_1 \circ x_2(y) \tag{18}$$

の導出を試みる．N_1 と上記 $x_{3,E}$ の導出後以下のように

$$\begin{aligned} x_{3,E} \circ N_1(y_A) &= x_{3,E} \circ N_1 \circ N_1^{-1} \circ x_4(x_A) \\ &= x_1 \circ x_2(x_A) = \kappa_A \end{aligned} \tag{19}$$

SSK を導出することができる．

2.3　研究手法

Bob の公開鍵 $y_{B,1}$ と $y_{B,2}$ が以下の要件を満たすとする．

要件 1　全ての $y \in \mathcal{S}$ に対して以下の計算

$$x_4^{-1} \circ N_1 \circ N_1^{-1} \circ x_4(y) = y \tag{20}$$

を現実的時間で行うことが困難，もしくは $y_{B,1}$ の逆写像, $x_4^{-1} \circ N_1$ を求めることが現実的時間で困難であること．

要件 2　(12) から N_1 を導出することが現実的時間内で困難，もしくは (18) を満たす写像 $x_{3,E}$ の導出が現実的時間で困難であること．

この時，要件 1 より Eve は x_A を (15) から現実的な時間で導出することができず，したがって (16) に進めない．また，要件 2 より明らかにパターン 2 の攻撃も現実的時間に行えない．注目すべきは，x_A に関する条件が上記 2 つの要件に含まれておらず，Bob の秘密鍵・公開鍵のみが大きく関わっている点にある．この考察から，Alice は自身の秘密鍵の鍵空間を，全探索攻撃が用意にならないように注意しながら (ビット数に関して) 小さくができるのではという仮説を得ることができる．もし可能であれば，Alice は鍵共有アルゴリズムの安全性を落とすことなく SSK 計算にかかるステップ数を下げることに繋がる可能性があると考え，本研究では以下の課題に取り組む．

課題 1　D-H を SAPKA の写像で記述し，非可換代数上に拡張し，N_1 に具体的な要素を入れ非対称性を持たせる [14].

課題 2　課題 1 で拡張したアルゴリズムにて Alice の秘密鍵空間を小さくした状態における，Eve の攻撃計算量を導出する [14].

課題 3　課題 2 で得た結果に対し，Alice の秘密鍵空間の大

きさについての下限と，下限における Alice の計算量を導出する．また，その際の計算速度を実装実験により検証する．

3. Diffie-Hellman アルゴリズムの拡張

$\mathcal{M}(d,\mathbb{Z}_p)$ を各要素が $\mathbb{Z}_p$ にある $d\times d$ 行列全体の集合とする (p は大きな素数)．スカラー $c\in\mathbb{Z}_p$，行列 $M\in\mathcal{M}(d,\mathbb{Z}_p)$ に対して以下の規則

$$c_{i,j}^{\circ M} := c^{M_{i,j}} \quad ; \quad i,j\in\{1,\cdots,d\} \tag{21}$$

に従う行列指数の累乗を用いて D-H を非可換代数 (行列) 上に拡張し，非対称性を持たせる．[14] で既に D-H の拡張を行なっており，ここでは拡張された D-H のアルゴリズムの説明を行う．

Step 1B　$\mathcal{S}:=\mathcal{M}(d,\mathbb{Z}_p)$ とし，Bob はまず 2 つの行列 x_B, N_B を $\mathcal{S}$ から選択し，$\mathbb{Z}_p$ の原始元 g を任意に選ぶ．その後 SAPKA における 5 つの写像を以下のように定める ($y\in\mathcal{S}, a,b\in\{1,\dots,d\}$)．

- $x_1(y)_{a,b} := \prod_{l\in\{1,\dots,d\}}(g^{\circ x_B})_{a,l}^{(y)_{l,g}}$
- $x_2 := id$
- $x_3(y)_{a,b} := \prod_{l\in\{1,\dots,d\}}(y)_{l,b}^{(x_B)_{a,l}}$
- $x_4(y) := g^{\circ y}$
- $N_1(y)_{a,b} := \prod_{l\in\{1,\dots,d\}}(y)_{l,b}^{(N_B^{-1})_{a,l}}$

ここで全ての $y\in\mathcal{S}, a,b\in\{1,\dots,d\}$ に対して

$$x_1\circ x_2(y)_{a,b} = (g^{\circ x_B y})_{a,b} = x_3\circ x_4(y)_{a,b} \tag{22}$$

が成立する．

Step 2B　Bob は公開鍵 $y_{B,1}$，$y_{B,2}$ を以下のように作成する ($a,b\in\{1,\dots,d\}$)．

$$y_{B,1}(y)_{a,b} := N_1^{-1}\circ x_4(y)_{a,b} = \prod_{l\in\{1,\dots,d\}}(g^{\circ y})_{l,b}^{(N_B)_{a,l}} \tag{23}$$

$$y_{B,2}(y)_{a,b} := x_1\circ x_2(y)_{a,b} = \prod_{l\in\{1,\dots,d\}}(g^{\circ x_B})_{a,l}^{(y)_{l,b}} \tag{24}$$

ここで (23) は以下のように書き直すことができる．

$$y_{B,1}(y)_{a,b} = \prod_{l\in\{1,\dots,d\}}(g^{\circ y})_{l,b}^{(N_B)_{a,l}} = \prod_{l\in\{1,\dots,d\}}(g^{\circ N_B})_{a,l}^{(y)_{l,b}} \tag{25}$$

したがって，Bob が Alice に $y_{B,1}$，$y_{B,2}$ を送るとは，写像 $y_{B,1}$，$y_{B,2}$ の計算規則を知らせ，行列 $g^{\circ N_B}, g^{\circ x_B}\in\mathcal{S}$ を送ることと同義である．

Step 1A　Alice は自身の秘密鍵として x_A を $\mathcal{S}$ から任意に選ぶ．

Step 2A　Alice は公開鍵 y_A を Bob の公開鍵 $y_{B,1}$ と自身の秘密鍵 x_A を用いて次のように計算する ($a,b\in\{1,\dots,d\}$)．

$$\begin{aligned} y_{Aa,b} := y_{B,1}(x_A)_{a,b} &= N_1^{-1}\circ x_4(x_A)_{a,b} \\ &= \prod_{l\in\{1,\dots,d\}}(g^{\circ N_B})_{a,l}^{(x_A)_{l,b}} \\ &= (g^{\circ N_B x_A})_{a,b} \end{aligned} \tag{26}$$

Step 3A　Alice は SSK を Bob の公開鍵 $y_{B,2}$ と自身の秘密鍵 x_A を用いて次のように計算する ($a,b\in\{1,\dots,d\}$)．

$$\begin{aligned} \kappa_{Aa,b} := y_{B,2}(x_A)_{a,b} &= x_1\circ x_2(x_A)_{a,b} \\ &= \prod_{l\in\{1,\dots,d\}}(g^{\circ x_B})_{a,l}^{(x_A)_{l,b}} \\ &= (g^{\circ x_B x_A})_{a,b} \end{aligned} \tag{27}$$

Step 3B　Bob は SSK を Alice の公開鍵 y_A を用いて次のように計算する ($a,b\in\{1,\dots,d\}$)．

$$\begin{aligned} \kappa_{Ba,b} := x_3\circ N_1(y_A)_{a,b} &= x_3\circ N_1\circ N_1^{-1}\circ x_4(x_A)_{a,b} \\ &= x_3\circ N_1(g^{\circ N_B x_A})_{a,b} \\ &= \prod_{l\in\{1,\dots,d\}}(g^{\circ N_B x_A})_{l,b}^{(x_B N_B^{-1})_{a,l}} \\ &= g^{\sum_{l\in\{1,\dots,d\}}(N_B x_A)_{l,b}(x_B N_B^{-1})_{a,l}} \\ &= g^{(x_B N_B^{-1} N_B x_A)_{a,b}} \\ &= g^{(x_B x_A)_{a,b}} = (g^{\circ x_B x_A})_{a,b} \end{aligned} \tag{28}$$

以下の図 2 は，拡張された D-H における各鍵共有プロセスの様子を表している．

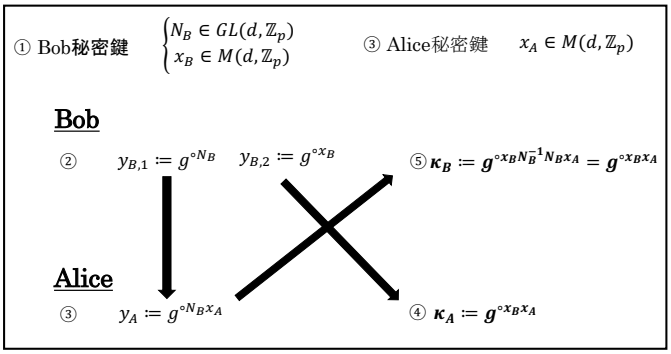

図 2　拡張された D-H の鍵共有プロセス
Fig. 2　Key Agreement Process of Matrix Type D-H

4. 拡張 D-H に対する攻撃

[14] で既に拡張 D-H に対する攻撃法と攻撃に要する計算量が考察されており，本章ではその結果を説明する．2.3 章で述べたように，まず Alice の秘密鍵空間をビット数に関して小さく定め，このとき Eve の計算量が鍵空間を小さくする前と比べてどう変化するかを考察する．以下では $\mathcal{S}$ の要素のビット数を関数 $|\cdot|:\mathcal{S}\to\mathbb{N}$ で表す．例えば $\mathcal{S}:=\mathcal{M}(d,\mathbb{Z}_p)$ とし，素数 p のビット数が $n\in\mathbb{N}$ である場合，任意の $y\in\mathcal{S}$ に対して，$|y|\le d^2 n$ となる．Eve の攻撃手法の考察の準備として以下を仮定する．

- p のビット数を $n\in\mathbb{N}$ とする．

- s を 1 次モニックでない多項式とし，ある $m \in \mathbb{N}$ に対して

$$n = s(m) > m \tag{29}$$

が成立するとする．
- $x_A \in \underline{\mathcal{S}} := \{y \in \mathcal{S} \ : \ |y| \leq d^2 m\} \subset \mathcal{S}$
- $x_B, N_B \in \mathcal{S}$

$\underline{\mathcal{S}}$ は各要素のビット数を (29) にしたがって小さくした行列の集合である．3 つめの仮定は Alice の秘密鍵 x_A はビット数に関して狭められた空間から選ばれることを指しており，Bob の秘密鍵 x_B，N_B はビット数に関して大きな集合から選ばれる．

Eve は 2.2 章で述べた 2 パターンの攻撃により SSK 導出を試みるとする．

4.1 パターン 1

Eve は Bob の公開鍵 $(a, b \in \{1, \ldots, d\})$
- $y_{B,1}(y)_{a,b} := N_1^{-1} \circ x_4(y)_{a,b} = \prod_{l \in \{1,\ldots,d\}} (g^{\circ y})_{l,b}^{(N_B)_{a,l}}$
- $y_{B,2}(y)_{a,b} := x_1 \circ x_2(y)_{a,b} = \prod_{l \in \{1,\ldots,d\}} (g^{\circ x_B})_{a,l}^{(y)_{l,b}}$

と公開情報
- $g^{\circ x_B}$
- $g^{\circ N_B}$
- $y_A = N_1^{-1} \circ x_4(x_A) = g^{\circ N_B x_A}$

より以下の式を計算することにより $x_A \in \underline{\mathcal{S}}$ の導出を試みる．

$$\begin{aligned} y_{B,1}^{-1}(y_A)_{a,b} &= x_4^{-1} \circ N_1 (g^{\circ N_B x_A})_{a,b} \\ &= \log_g (g^{\circ N_B^{-1} N_B x_A})_{a,b} \\ &= \log_g (g^{x_A})_{a,b} = (x_A)_{a,b} \end{aligned} \tag{30}$$

(30) にあるような指数が行列 $M \in \mathcal{M}(d, \mathbb{Z}_p)$ の場合の離散対数問題は以降，次のように表す．

$$\log_g M = (\log_g M_{a,b}) \qquad (g \in \mathbb{Z}_p) \tag{31}$$

Eve は (30) における 2 つめの等式に進むために，Bob のみが知っている写像 N_1 を導出しなければならない．N_1 の導出は，以下の離散対数問題

$$\log_g g^{\circ N_B} = N_B \tag{32}$$

を解くことで行列 N_B を導出することで十分である．[15] などにより有限体上の離散対数問題は, 離散対数の探索範囲のビット数の準指数オーダーで表されることが知られており，(32) を解くのに必要な計算量オーダーは $O(d^2 2^{\frac{n}{2}}) = O(2^{\frac{n}{2}})$ に従う．(30) の 3 つ目の等式では同様に d^2 回の離散対数問題を解く必要があるが，$x_A \in \underline{\mathcal{S}}$ よりこのときの計算量オーダーは $O(2^{\frac{m}{2}})$ に従い，(29) より (32) の場合より軽い計算で済む．[14] の Theorem 1 では，(32) を解く問題は y_A から x_A を導出する問題に帰着することが示されており，(32) を解く問題の計算量が (30) の 3 つめの計算量以下になる，すなわち (29) が成立するように n, m が選ばれていない場合は明らかに拡張 D-H に対する攻撃計算量は Alice の秘密鍵空間 $\underline{\mathcal{S}}$ に依存する．

4.2 パターン 2

Eve は 4.1 章と同様に写像 N_1 を導出しなければならない．この際必要な計算量オーダーは，4.1 章と同様に $O(2^{\frac{n}{2}})$ に従う．

4.3 考察

4.1 章より，パターン 1 の攻撃計算量が Alice の秘密鍵空間 $\underline{\mathcal{S}}$ と関連しないための必要十分条件は (29) が成立し，$N_B \in \mathcal{S}$ であることが言える．また，4.2 章における攻撃は Alice の秘密鍵空間 $\underline{\mathcal{S}}$ とは関連せず，Bob の N_B，x_B のみに依存することから拡張 D-H に対する攻撃計算量が Alice の秘密鍵空間と関連しないための必要十分条件は，そのまま「(29) が成立し，$N_B \in \mathcal{S}$」，ということが言える．以上により，Alice の秘密鍵空間は任意に小さくしても良いと考えられるかもしれないが，上や [14] では
- Eve による x_A に対する全探索攻撃
- $O(2^{\frac{n}{2}})$ より大きいが，全探索攻撃以下の計算量を要する攻撃アルゴリズムの存在可能性

についての議論を行なっていないことに注意しなくてはならない．2 つめの点について，ある暗号アルゴリズムに対する有効な攻撃法がないことを証明することが困難であることと同様に，$O(2^{\frac{n}{2}})$ より大きいが，全探索攻撃以下の計算量を要する攻撃アルゴリズムがないと証明することは難しい．現状拡張 D-H に対してそのような攻撃法は見つかっていないことから，本稿ではそのような攻撃法が存在しないと仮定して議論を進める．x_A に対する全探索攻撃に要する計算量オーダーは，$x_A \in \underline{\mathcal{S}}$ であることと，全探索攻撃の解は各列ベクトルごとに独立であることに注意すると，$O(d2^{dm}) = O(2^{dm})$ である．

このとき Alice は全探索攻撃にかかる計算量が，(32) を解くのに要する計算量以下にならない程度まで秘密鍵空間を小さくできると言える．すなわち，

$$2^{dm} = 2^{\frac{n}{2}} = 2^{\frac{s(m)}{2}} \tag{33}$$

が恒等的に成り立つような n, m，つまり

$$n = s(m) = 2dm \tag{34}$$

の場合，全探索攻撃にかかる計算量は (32) を解くのに要する計算量と同等になり，例えば有限体上の離散対数問題が困難になる程度に n が大きければ，全探索攻撃も同様に困難となる．

[14] では，条件 (34) 下における Alice と Bob が SSK 計算に必要な計算量 (計算ステップ数) がビット数 m の関数 $T_A, T_B : \mathbb{N} \to \mathbb{N}$ で以下のように

$$T_A(m) := 2d^3m \tag{35}$$

$$T_B(m) := 4d^3m + 2d^4m \tag{36}$$

導出されており，これらを d の関数とみなすと Alice の計算量オーダーは Bob のそれよりも低くなる．以下の章では，Alice と Bob の計算速度を比較したり，拡張 D-H の Alice の計算速度を D-H における Alice の計算速度と比較するなどして，拡張 D-H の性能・特徴を実装実験により評価する．また，[14] では x_A の空間として行列の代わりにベクトル空間を選べることが示されており，このとき Alice，Bob の計算量は $T_A(m) := 2d^2m$，$T_B(m) := 6d^3m$ となり行列の場合より下げることができる．安全性も Alice の秘密鍵空間が行列の場合と同等であることが，[14] の Section 4.3 における攻撃手法を Alice の秘密鍵空間がベクトル空間の場合にも同様に適用できることから，容易に示せる．以下では Alice の秘密鍵空間をベクトル空間として実装実験を行う．

5. 実装実験

以下では，D-H および拡張 D-H の計算速度を計測し性能を評価する．5.1，5.2 章では以下の環境にて実装実験を行う．

- OS: Windows 10 Pro
- Processor: 2.90 GHz Intel Core i7-10700
- RAM: 8 GB
- Language: JAVA

5.1 拡張 D-H における Alice と Bob の計算速度の比較

本章では，素数のビット数 n を固定し，d, m を変動させて Alice と Bob の計算速度を計測する．n がそれぞれ 1024, 2048, 3072, 4096, 5120 の場合の計算速度を 100 回計測し，平均したものを以下の表 1 にまとめている．なお，d, m, n の関係は (34) にしたがっており，拡張 D-H におけるベクトルの各要素は独立して計算できるため，Java 向け並列計算用フリーソフト omp4j(http://www.omp4j.org/) を用いて並列で計算を行なっている．

表 1 拡張 D-H における Alice と Bob の計算速度の比較
Table 1 Comparison of Calculation Speed between Alice and Bob

n	m	d	Alice 計算時間 (ms)	Bob 計算時間 (ms)
1024	256	2	2.049359	4.61656
1024	128	4	2.907048	9.564223
2048	512	2	6.145738	24.649098
2048	256	4	5.670952	47.46677
3072	768	2	16.041431	83.153447
3072	384	4	16.275069	155.774181
4096	1024	2	31.631416	186.14481
4096	512	4	34.878155	362.323917
5120	1280	2	61.043727	337.193849
5120	640	4	64.028805	688.312635

$x_B, N_B \in \mathcal{S}$ より Bob の計算速度は m によらず d, n のみに依存することが予想でき，実際に表 1 では同サイズの n に対して，d が大きくなると計算速度も大きくなっていることが確認できる．Alice の計算速度が d に応じて大きく変化しない，あるいは $n = 2048$ の場合のように $d = 4$ のときの方が $d = 2$ のときより高速になることがあるのは，d に応じて m が減少することによりベクトル各要素での累乗計算が軽量化されたことと並列計算によるものだと考えられる．

d を大きくすることで Bob における計算速度が上昇することを考慮すると $d = 2$ に固定した方が良いと考えられるが，d と安全性の関連性は未だ明らかになっていないため，今後検証が必要である．

5.2 拡張 D-H の Alice と D-H の Alice の計算速度の比較

表 2 は各素数ビット n における D-H の Alice・Bob の計算速度を表した表である (5.1 と同様に 100 回平均)．表 1 と比べると，拡張 D-H における Alice の計算速度は同素数ビット n の D-H の Alice の速度より 2 から 3 倍近く高速であり，拡張 D-H の Bob の速度は，D-H の Bob より 2 倍近く遅いことが確認できる．同じ素数ビット n における拡張 D-H と D-H の安全性は，(4.3 章の仮定の上) 等しいと考えられるので，これらの結果から D-H を SAPKA を用いて非可換代数上に拡張することの利点が説明できる．

表 2 D-H における Alice と Bob の計算速度
Table 2 Comparison of Calculation Speed between Alice and Bob in D-H

n	計算時間 (Alice, ms)	計算時間 (Bob, ms)
1024	2.30328909	2.37211636
2048	14.5662618	14.6508582
3072	47.2379936	47.44371
4096	105.351657	105.796802
5120	194.378962	194.523875

5.3 異なるデバイス間での鍵共有に関する考察

Bob が計算能力の高いデバイスを，Alice が計算能力のより低いデバイスをそれぞれ用いて鍵共有を行うことを想定した考察を行う．ただし以下では，公開鍵の送信，素数生成に要する時間は考慮しないこととする．

D-H において，Alice と Bob はそれぞれ独立で公開鍵を計算することができ，SSK は自身の公開鍵の計算が終わり，かつ通信相手の公開鍵が到着次第計算される．したがって，D-H において Alice と Bob が SSK を導出するのに必要な計算時間は以下のように与えられる．

$$max(T(y_A), T(y_B)) + max(T(\kappa_A), T(\kappa_B)) \tag{37}$$

ここで T は各鍵の計算に要する時間を表す．

拡張 D-H においては，まず Bob が公開鍵 $y_{B,1}$ を計算し

なくてはならない．Alice は $y_{B,1}$ の到着後，自身の公開鍵 y_A を計算し，Bob は $y_{B,1}$ の計算後直ちにもう一つの公開鍵 $y_{B,2}$ の計算を行うため，Alice の y_A と Bob の $y_{B,2}$ はそれぞれ独立して計算が行われる．Alice の SSK は自身の y_A の計算が終わり，$y_{B,2}$ が Bob から届いた後計算され，Bob の SSK は $y_{B,2}$ の計算が終わり y_A が Alice から届き次第計算されることから，拡張 D-H において Alice と Bob が SSK を導出するのに必要な計算時間は以下のようになる．

$$T(y_{B,1}) + max(T(y_A), T(y_{B,2})) + max(T(\kappa_A), T(\kappa_B)) \quad (38)$$

Alice の計算が上記のような一般的な環境で行われているとし，Bob の計算は各鍵の計算時間が無視できるほど小さくなるような環境で行われているとすると，Alice と Bob が SSK を計算し終わるまでにかかる時間は結局 Alice の計算時間に依存する．本紀要で使用した環境を Alice のデバイスとして鍵共有を行う場合，5.1，5.2 章の結果を用いると結局，拡張 D-H は D-H の 3 倍程度高速に鍵共有を行えることが言える．高い計算能力を持つ大企業・政府等のサーバーが Bob の計算を担い，Alice の計算を一般ユーザーの PC が担うなどの環境下で，拡張 D-H は安全・高速な通信に貢献できると考えられる．

6. 終わりに

本研究では計算能力が限られたデバイスを用いた安全な鍵共有を行うことが困難になる可能性があるという問題に対して，SAPKA フレームワークを用いて D-H を拡張することで，能力が低いデバイスにおける計算処理を軽量化できるアルゴリズム (拡張 D-H) を考案した．また簡易的な実装実験により，拡張 D-H における Alice と Bob の計算時間が非対称であること，Alice の計算時間が D-H と比較して高速であることが判明した．今後の課題として以下が挙げられる．

- [14] における SAPKA サブクラスを用いて，楕円曲線 D-H や耐量子鍵共有アルゴリズムに対して，D-H と同様の拡張が行えるかどうか確認し，可能であれば Alice の鍵空間を小さくした上での攻撃計算量の導出や実装実験による性能の評価を行う．
- 拡張 D-H に対する，離散対数問題を用いない攻撃法の考察をより深め，4.3 章における考察が妥当なものであるかを再度確認する必要がある．
- 拡張 D-H は従来の D-H と比較して公開鍵の鍵長が明らかに長いため，この点に関する通信への影響を考察しなければならない．
- 中間者攻撃に対する考察を一切行なっていないため，従来の電子署名アルゴリズムとの適応性などの考察を行わなくてはならない．

以上の課題の解決により，計算能力が限られたデバイス上での安全かつ高速な鍵共有の実現に寄与できると考える．

参考文献

[1] Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978** *21*, pp. 120–126.
[2] Diffie, W.; Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, pp. 644–654.
[3] Adrian, D.; Bhargavan, K.; Durumeric, Z.; Gaudry, P.; Green, M.; Halderman, J.A.; Heninger, N.; Springall, D.; Thomé, E.; Valenta, L.; et al. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 5–17.
[4] 情報処理推進機構; 情報通信研究機構, TLS 暗号設定ガイドライン, July, 2020, https://www.ipa.go.jp/security/ipg/documents/ipa-cryptrec-gl-3001-3.0.1.pdf. (Acceced on 4 August 2021)
[5] Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, pp. 1484–1509.
[6] Ajtai, M.; Dwork, C. A public-key cryptosystem with worst-case/average-case equivalence. IIn Proceedings of the 50th ACM Symposium on Theory of Computing, El Paso, TX, USA, 4–6 May 1997, pp. 284–293.
[7] Khot, S., Hardness of approximating the shortest vector problem in lattices. Journal of the ACM (JACM) **2005**, *52(5)*, pp. 789–808.
[8] Lyubashevsky, V.; Peikert, C., and Regev, O. On ideal lattices and learning with errors over rings, In Annual international conference on the theory and applications of cryptographic techniques, Springer, Berlin, Heidelberg, May, 2010, pp. 1–23.
[9] Hoffstein, J; Pipher, J; Silverman, J,H. NTRU: A Ring Based Public Key Cryptosystem. In Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, J.P. Buhler (ed.), Lecture Notes in Computer Science 1423, Springer-Verlag, Berlin, 1998, pp. 267–288.
[10] Alkim, E.; Ducas, L.; Poppelmann, T.; Schwabe, P. Post-quantum key exchange—A new hope. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10-12 August, 2016, pp. 327–343.
[11] Coppersmith, D.; Shamir, A. Lattice Attacks on NTRU. Advances in Cryptology—EUROCRYPT '97, Lecture Notes in Computer Science; Fumy, W., Ed., Springer: Berlin, Germany, 1997, Volume 1233.
[12] Laine, K.; Lauter, K. Key Recovery for LWE in Polynomial Time. IACR Cryptology ePrint Archive 2015, no. 176. Available online: https://eprint.iacr.org/2015/176.pdf (accessed on 5 August 2021).
[13] Accardi, L.; Iriyama, S.; Regoli, M.; Ohya, M. Strongly Asymmetric Public Key Agreement Algorithms; Technical Report ISEC2011-20; IEICE: Tokyo, Japan, 2011; pp. 115–121.
[14] Iriyama, S.; Jimbo, K.; Regoli, M. New Subclass Framework and Concrete Examples of Strongly Asymmetric Public Key Agreement. Applied Sciences, 2021, Vol.11, No.12, 5540.
[15] Pollard, J. Monte Carlo Methods for Index Computation (*mod p*). Math. Comput. 1978, 32, pp. 918–924.