2枚の追加カードを用いた多入力ANDプロトコルのシャッフル回数の削減

中林 佳祐[1,a)]　宮原 大輝[1,2]　水木 敬明[1,2]

概要：カードベース暗号は，物理的なカード組を用いて秘密計算を実現する暗号プロトコルである．本稿では，黒と赤のカードを 1 枚ずつ使用して 1 ビットを表す符号化ルールを対象にする．その符号化ルールに基づいてあるビットを表している 2 枚の裏になったカード列は，コミットメントと呼ばれる．2 つのコミットメントが与えられたときに，それらの論理積を表すコミットメントを出力するプロトコルを 2 入力コミット型 AND プロトコルといい，最も効率的なものは，2009 年に Mizuki と Sone によって提案された追加カード 2 枚と 1 回のシャッフルを用いるものである．n 入力の論理積のコミットメントを得るように拡張する場合は，このプロトコルを $n-1$ 回繰り返せばよく，その際に必要なカードは $2n+2$ 枚，シャッフルは $n-1$ 回となる．本稿では，追加カードを 2 枚だけ用いてよいという条件の下，n 入力の論理積の秘密計算に必要なシャッフル回数の削減を行う．具体的には，2020 年に Shinagawa と Nuida によって提案された Batching と呼ばれる技術を応用すると，必要なカード枚数は $2n+2$ 枚のままで，20 以上の n についてシャッフル回数を削減できることを報告する．

キーワード：カードベース暗号, カード, 秘密計算, コミット型, 論理積

Reducing the Number of Shuffles for a Multi-input AND Protocol Using Two Additional Cards

Keisuke Nakabayashi[1,a)]　Daiki Miyahara[1,2]　Takaaki Mizuki[1,2]

Abstract: Card-based cryptography is a cryptographic protocol realizing secure computations using a deck of physical cards. This study deals with a typical encoding rule: a bit is represented as the order of a pair of a black card and red card. Such a pair of two face-down cards representing a bit is called a commitment. A two-input committed-format AND protocol takes two commitments as input and produces a commitment to the logical AND of the inputs. The most efficient protocol is the one proposed by Mizuki and Sone in 2009, which uses two additional cards and one shuffle. By repeating the execution of that protocol $n-1$ times, the logical AND of n inputs can be obtained, where $2n+2$ cards and $n-1$ shuffles are used. In this study, we reduce the number of shuffles required for performing a secure computation of the logical AND of n inputs under such a condition that two additional cards are given. Precisely, we employ the technique called batching proposed by Shinagawa and Nuida in 2020. This technique allows us to reduce the number of shuffles for more than 20 inputs under the condition.

Keywords: Card-based cryptography, Deck of cards, Secure computation, Committed format, Logical AND function

1. はじめに

物理的なカード組を用いて秘密計算を行う手法を**カードベース暗号**という．本稿では裏面が $\boxed{?}$ のように区別がつ

1 東北大学
Tohoku University
2 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST)
a) nakabayashi.keisuke.r2@dc.tohoku.ac.jp

© 2021 Information Processing Society of Japan

かず，表面は♣または♡のいずれかである 2 種類のカードを用いる．これらのカードを次のように置くことでブール値を表す．

$$\boxed{\clubsuit}\boxed{\heartsuit} = 0,\ \boxed{\heartsuit}\boxed{\clubsuit} = 1$$

この符号化ルールに従って裏向きに置かれた 2 枚のカードが，あるビット $x \in \{0,1\}$ を表すとき，この 2 枚のカードを x の**コミットメント**と呼び，次のように表す．

$$\underbrace{\boxed{?}\boxed{?}}_{x}$$

2 入力 AND プロトコルは，2 つのコミットメント（$x, y \in \{0,1\}$ のコミットメント）を入力とし，シャッフルや並べ替え，カードをめくる操作によって，論理積 $x \wedge y$ の秘密計算を行う．出力が $x \wedge y$ のコミットメントとして得られるとき，そのようなプロトコルを**コミット型**プロトコルという．

$$\underbrace{\boxed{?}\boxed{?}}_{x}\underbrace{\boxed{?}\boxed{?}}_{y} \to \cdots \to \underbrace{\boxed{?}\boxed{?}}_{x \wedge y}$$

1.1 既存の 2 入力コミット型 AND プロトコル

既存の 2 入力コミット型 AND プロトコルの中で最も効率的なものは，2009 年に Mizuki と Sone [4] によって提案されたプロトコルである．これは，x と y のコミットメントが与えられたときに，2 枚の追加カードと 1 回のランダム二等分割カット（RBC）というシャッフル操作を用いて，$x \wedge y$ のコミットメントを出力する．本稿ではこれを *MS-AND* **プロトコル**と呼ぶ．

$$\underbrace{\boxed{?}\boxed{?}}_{x}\underbrace{\boxed{?}\boxed{?}}_{y}\boxed{\clubsuit}\boxed{\heartsuit} \to \overset{\text{RBC}}{\cdots} \to \underbrace{\boxed{?}\boxed{?}}_{x \wedge y}$$

1.2 既存の多入力コミット型 AND プロトコル

MS-AND プロトコルはコミット型であるため，その出力を別のプロトコルに入力することで多入力の論理関数を計算することができる．$n(\geq 3)$ 入力の論理積のコミットメントを得るには，MS-AND を $n-1$ 回繰り返せばよい．

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_n}\boxed{\clubsuit}\boxed{\heartsuit} \to \overset{n-1\ \text{MS-AND}}{\cdots} \to \underbrace{\boxed{?}\boxed{?}}_{x_1 \wedge x_2 \wedge \cdots \wedge x_n}$$

具体的には次のようにランダム二等分割カットを $n-1$ 回用いる．2 枚の追加カードは再利用することができる．本稿ではこれを n 入力 MS-AND プロトコルと呼ぶ．

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\boxed{\clubsuit}\boxed{\heartsuit} \to \overset{\text{RBC}}{\cdots} \to \underbrace{\boxed{?}\boxed{?}}_{x_1 \wedge x_2}$$

$$\underbrace{\boxed{?}\boxed{?}}_{x_1 \wedge x_2}\underbrace{\boxed{?}\boxed{?}}_{x_3}\boxed{\clubsuit}\boxed{\heartsuit} \to \overset{\text{RBC}}{\cdots} \to \underbrace{\boxed{?}\boxed{?}}_{x_1 \wedge x_2 \wedge x_3}$$

$$\vdots$$

$$\underbrace{\boxed{?}\boxed{?}}_{x_1 \wedge x_2 \wedge \cdots \wedge x_{n-1}}\quad\underbrace{\boxed{?}\boxed{?}}_{x_n}\boxed{\clubsuit}\boxed{\heartsuit} \to \overset{\text{RBC}}{\cdots} \to \underbrace{\boxed{?}\boxed{?}}_{x_1 \wedge x_2 \wedge \cdots \wedge x_n}$$

1.3 本稿の貢献

本稿では追加カードを 2 枚だけ用いるという条件の下，n 入力のコミット型 AND プロトコルに必要なシャッフル回数の削減を行う．すなわち，既存方式で最も効率の良い n 入力 MS-AND プロトコルを改良する．具体的には，n 入力に拡張した MS-AND プロトコルに，2020 年に Shinagawa と Nuida [5] によって提案された Batching と呼ばれる技術を適用することで，必要なカード枚数は $2n+2$ 枚のままで $n \geq 20$ なる n について，シャッフル回数を削減したプロトコルを新たに提案する．

2. 準備

本節ではまず，カードベース暗号で使用される操作について説明する．次に，既存研究の詳細な説明，Batching についての説明を行う．

2.1 カードベース暗号で使用される操作

カードベース暗号では，カード列に対して主に 3 つの操作を行う [2,3]．ここでは ℓ 枚のカード列を考える．

並べ替え　S_ℓ を ℓ 次の対称群とし，カード列に対し，ある置換 $\pi \in S_\ell$ を適用する．この操作を (perm, π) と書く．

$$\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\cdots\overset{\ell}{\boxed{?}} \xrightarrow{(\mathsf{perm},\pi)} \overset{\pi^{-1}(1)}{\boxed{?}}\overset{\pi^{-1}(2)}{\boxed{?}}\cdots\overset{\pi^{-1}(\ell)}{\boxed{?}}$$

めくる　カード列の左から t 枚目のカードをめくってカードの色を確認する．この操作を $(\mathsf{turn}, \{t\})$ と書く．

$$\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\cdots\overset{t}{\boxed{?}}\cdots\overset{\ell}{\boxed{?}} \xrightarrow{(\mathsf{turn},\{t\})} \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\cdots\overset{t}{\boxed{\clubsuit}}\cdots\overset{\ell}{\boxed{?}}$$

シャッフル　カード列に対し，ある置換集合 $\Pi \subseteq S_\ell$ から一様ランダムに選ばれた置換 $\pi \in \Pi$ を適用する．この操作を (shuf, Π) と書く．

$$\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\cdots\overset{\ell}{\boxed{?}} \xrightarrow{(\mathsf{shuf},\Pi,)} \overset{\pi^{-1}(1)}{\boxed{?}}\overset{\pi^{-1}(2)}{\boxed{?}}\cdots\overset{\pi^{-1}(\ell)}{\boxed{?}}$$

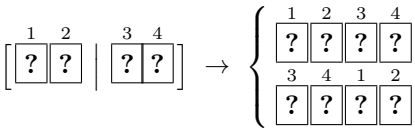
ただし，Π に含まれる置換の内, どの置換が適用されたのかは誰も知り得ない．

任意の $\tau, \sigma \in \Pi$ に対して $\tau\sigma \in \Pi$ であるとき，そのシャッフルを closed である（閉じている）という．こ

の条件を満たさないとき，non-closed である（閉じていない）という．

2.2 ランダム二等分割カット

ランダム二等分割カットは 2009 年に Mizuki と Sone [4] によって考案されたシャッフル操作である．これは，カード列を二等分にし，その左右をランダムに入れ替えるシャッフル操作である．カード列にランダム二等分割カットを適用するとき，$[\cdot|\cdot]$ と表記する．例として，4 枚のカード列にランダム二等分割カットを適用すると，次のカード列のいずれかが 1/2 の確率で得られる．



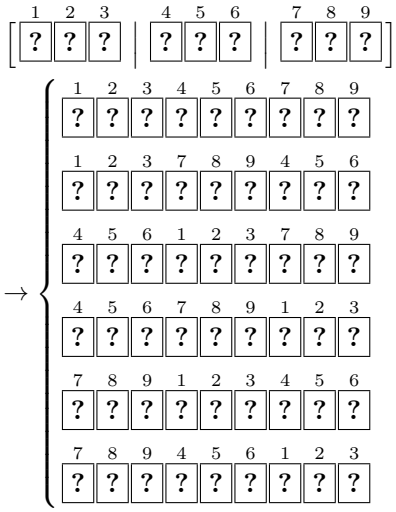
このランダム二等分割カットは次のように書くことができる．

$$(\mathsf{shuf}, \{\mathsf{id}, (1\,3)(2\,4)\})$$

ランダム二等分割カットは身近な道具を用いて安全に実装できることが知られている [6]．

2.3 Pile-scramble shuffle

Pile-scramble Shuffle [1] は一様で閉じているシャッフルである．このシャッフルは，カード列を同じ枚数の複数の組に分け（それぞれを Pile と呼ぶ），それらの順番のみをランダムに入れ替えるシャッフルである．カード列に Pile-scramble shuffle を適用するとき，$[\cdot|\cdots|\cdot]$ と表記する．例として，3 つの Pile としての 9 枚のカード列に Pile-scramble shuffle を適用すると，次のカード列のいずれかが 1/6 の確率で得られる．

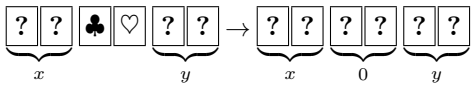


Pile-scramble shuffle はそれぞれの Pile を封筒に入れてシャッフルすることで簡単に実装できる．また，ランダム二等分割カットは 2 つの Pile に対する Pile-scramble shuffle であるといえる．

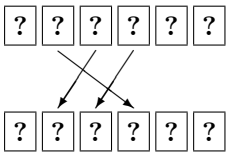
2.4 MS-AND プロトコル

MS-AND プロトコルは 2009 年に Mizuki と Sone [4] によって提案された 2 入力コミット型 AND プロトコルであり，x のコミットメントと y のコミットメント（$x, y \in \{0, 1\}$）の 2 つが与えられたときに，2 枚の追加カードと 1 回のランダム二等分割カット（RBC）を用いて $x \wedge y$ のコミットメントを出力するプロトコルである．その手順を以下に示す．

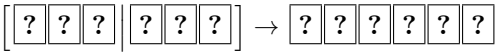
(1) 2 つの入力のコミットメントと 2 枚の追加カードを次のように置き，中央の 2 枚を裏返す．



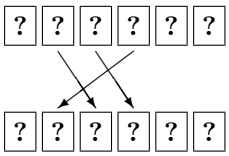
(2) カード列を次のように並び替える．



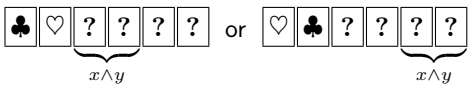
(3) カード列に対してランダム二等分割カットを適用する．



(4) カード列を次のように並び替える．



(5) 左から 2 枚のカードをめくる．めくられた 2 枚のカード列の並びに応じて次のように $x \wedge y$ のコミットメントが得られる．



多入力へ拡張する場合はここで得られた出力のコミットメントを次の入力として利用する．また，めくった 2 枚のカードは次に計算を行うときの追加カードとして再利用できる．これらの 4 枚のカードとは別に残った 2 枚のカードは，$\overline{x} \wedge y$ のコミットメントとなっており，情報が漏れないようシャッフルしてからめくることで再利用できる．

2.5 Batching

Batching [5] とは，並列に実行可能な複数の Pile-scramble shuffle を 1 つの Pile-scramble shuffle へ変換する技術のことである．そのときに追加カードを用いてどの Pile-scramble shuffle かを識別可能にする必要がある．

N 個の Pile-scrable shuffle を Batching するとき，それ

ぞれの Pile-scramble shuffle を識別するには N 種類の符号を用意する必要がある．例えば，$\boxed{\clubsuit} = 0$, $\boxed{\heartsuit} = 1$ とし，2 進数として扱うことで $\lceil \log N \rceil$ 枚のカードで N 種類の符号を構成することができる．したがって，i 番目の Pile-scramble shuffle が l_i 個の Pile を含んでいるとすると，$\sum_{i=1}^{N} l_i \cdot \lceil \log N \rceil$ 枚の追加カードが必要となる．

例として，Batching を用いて，MS-AND で行うランダム二等分割カットを 2 つ並列に行う手順を示す．すなわち，

$$\left[\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{?}\boxed{?}\boxed{?}\right], \left[\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{?}\boxed{?}\boxed{?}\right]$$

を 1 回のシャッフルで行いたい．

(1) 各 Pile の先頭に，識別のための追加カードを挿入し，1 つの Pile-scramble shuffle へ変換する．

$$\left[\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\right]$$

(2) 追加したカードを裏返し, Pile-scramble shuffle を行う．

$$\left[\boxed{?}\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{?}\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{?}\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{?}\boxed{?}\boxed{?}\boxed{?}\right]$$
$$\rightarrow \boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}$$

(3) 各 Pile へ追加したカードの部分を表向きにする．例として次のカード列が得られたとする．

$$\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}$$

(4) ステップ 1 で追加カードを挿入したときと同じように，左に $\boxed{\clubsuit}$ が追加された Pile を，右に $\boxed{\heartsuit}$ が追加された Pile となるように並べ替える．上の例では，4 番目の Pile を 2 番目の Pile の前へ移動する．

$$\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}$$
$$\rightarrow \boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}$$

(5) 表向きにした追加カードを取り除けば，2 つの Pile-scramble shuffle を 1 つの Pile-scramble shuffle で行えたことになる．

3. 提案プロトコル

本節では 1.2 節で紹介したコミット型の多入力 AND プロトコルへ，2.5 節の Batching を応用することで，シャッフル回数を削減したコミット型の多入力 AND プロトコルを提案する．

3.1 提案プロトコルのアイデア

2.5 節で述べたように，Batching を用いることで複数の Pile-scramble shuffle を 1 つの Pile-scramble shuffle へ変換することができる．これを多入力 AND プロトコルへ適用し，多入力 AND 計算を行うときのシャッフル操作をいくつかまとめて行うことで，シャッフル回数を削減することができる．しかし，追加カードを用意しなければ Batching を適用することができない．そこで，2.4 節で述べたように，AND 計算を行ったときに生じるフリーカードを，Batching を適用するときの追加カードとして再利用することで，合計カード枚数を増やすことなくシャッフル回数を削減する．

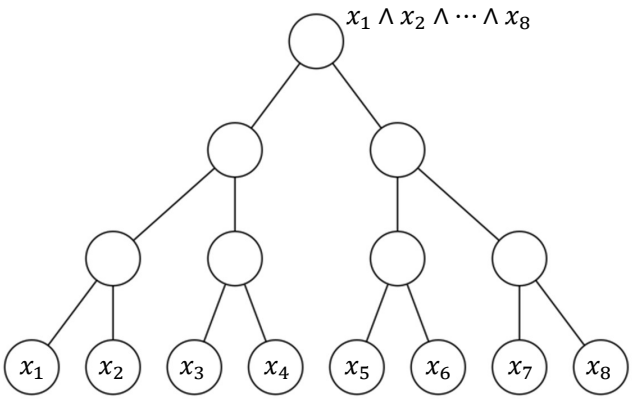


図 1 2^3 入力の MS-AND プロトコルの Batching

よって，入力の内，あるところまでは既存方式と同様に MS-AND プロトコルを繰り返し行い，フリーカードが Batching を適用するのに十分にたまった時点で，残りの部分に対しては MS-AND プロトコルのシャッフルに Batching を適用することで MS-AND プロトコルを並列に行い，シャッフル回数を削減する．すなわち，ある m に対して，1 から m 番目のコミットメントについては MS-AND プロトコルを直列に実行し，$m+1$ 番目以降は並列に実行する．

$$\underbrace{\boxed{?}\boxed{?}\cdots\boxed{?}\boxed{?}}_{x_1 \quad x_m}\qquad\qquad \underbrace{\boxed{?}\boxed{?}\cdots\boxed{?}\boxed{?}}_{x_{m+1} \quad x_n}$$

MS-AND プロトコルを繰り返し行う　MS-AND プロトコルを並列に行う

3.2 Batching を用いた MS-AND プロトコル

3.1 節で述べたように，本稿では MS-AND プロトコルに Batching を適用する．まず，入力が $k \geq 3$ なる 2^k 個のコミットメントである場合についてどのように Batching を適用するかについて説明し，次に $j \geq 2$ とするときに入力が $j2^k$ 個のコミットメントの場合について説明する．

3.2.1 2^k 入力の場合

2^k 個のコミットメントから 2^{k-1} 個のペアを作り，各ペアで MS-AND プロトコルを実行することを考え，その際に Batching を適用する．得られた 2^{k-1} 個のコミットメントに対してまた同様に Batching を適用していく．これを **2^k-Batching** と呼ぶ．このときの Batching を適用する過程は深さが k である 1 つの完全二分木で表すことができ，2^3 入力の場合は図 1 のようになる．また，シャッフル回数は木の深さと同じになるため，k 回となる．

次に，このときに必要な追加カードの枚数について考

える．MS-AND プロトコルに Batching を適用するときの追加カードは，Batching で先頭に追加するカードと，MS-AND プロトコル自体の追加カードの 2 つが必要となる．まず，前者の Batching で先頭に追加するカードについて述べる．いま，入力数は 2^k であるので，必要なランダム二等分割カットの数は 2^{k-1} 回である．ランダム二等分割カットは 2 個の Pile を含んでいるので 2.5 節の式により，$2^{k-1} \times 2 \times \log 2^{k-1} = 2^k(k-1)$ 枚の追加カードが必要となる．次に，MS-AND プロトコル自体の追加カードであるが，これは 2 つの入力に対し 2 枚必要なので，$2^k/2 \times 2 = 2^k$ 枚となる．したがって，追加カードの合計は $2^k(k-1) + 2^k = k2^k$ 枚となる．

3.2.2 $j2^k$ 入力の場合

この場合は j 個の完全二分木があると言い換えることができるので，まず，この j 個の木を 1 つにまとめる作業を行う．2 つの木について，それぞれの 2^k 個のコミットメントに Batching を適用して得られる 2^{k-1} 個のコミットメントを集めると，新たに 2^k 個のコミットメントが得られる．それらと別の木に対応している 2^k 個のコミットメントに対して同様の操作を行うと，再び新たな 2^k 個のコミットメントが得られる．これを繰り返すことで，$j2^k$ 個のコミットメントを 2^k 個のコミットメントにまとめることができる．すなわち，j 個の木が 1 つにまとめられる．以降ではこの 1 つの木にまとめる操作を **$j2^k$-Batching** と呼ぶ．この木に対して 2^k-Batching を適用することで，目的のコミットメントを得ることができる．

2 つの木を 1 つに変換するには 2^k 個のコミットメントから 2^{k-1} 個のコミットメントを得る Batching を 2 回行うため，2 回のシャッフルが必要となる．したがって，合計で $2(j-1)$ 回のシャッフルによって j 個の木を 1 つにまとめることができる．

3.3 フリーカードと最大 Batching

MS-AND プロトコルを行ったときに生じるフリーカードの数と Batching に必要な追加カードの関係から，どれくらい MS-AND プロトコル繰り返せば Batching を適用するのに充分なフリーカードが集められるのかを考える．

n 個のコミットメントと 2 枚の追加カードが与えられたとき，2^k-Batching を適用したいとし，$n - 2^k$ 個のコミットメントには直列で MS-AND プロトコルを適用するとしよう．フリーカードは MS-AND プロトコルを 1 回行うたびに 2 枚生じるので，MS-AND プロトコルの $n - 2^k - 1$ 回の直列実行で，$2(n - 2^k - 1) = 2n - 2^{k+1} - 2$ 枚のフリーカードが生じる．また，元々の 2 枚の追加カードもフリーカードとなるため，これを加えて，フリーカードの合計は $2n - 2^{k+1}$ 枚となる．前述の通り，2^k-Batching に必要な追加カードは $k2^k$ であるので，2^k-Batching の適用のためにはフリーカードの枚数がこれ以上である必要があり，$2n - 2^{k+1} \geq k2^k$ である．従って，この条件を満たす n と k に対してのみ，以上のことが実行できる．

そこで，n が与えられたとき，2^k-Batching の適用が可能な最大の k を $\mathsf{MBS}(n)$ と書くことにし（Max Batch Size から命名），次のように定義できる．

$$\mathsf{MBS}(n) \stackrel{\text{def}}{=} \max\{k \mid \underbrace{2n - 2^{k+1}}_{\text{フリーカードの数}} \geq \underbrace{k2^k}_{\text{追加カードの数}}\}$$

表 1 に n と $\mathsf{MBS}(n)$ の関係を示した．例えば，入力数 n が $20 \leq n \leq 47$ のときには 2^3-Batching が適用可能であることを示している．

表 1 $\mathsf{MBS}(n)$ の値

n	$\mathsf{MBS}(n)$
$20 \leq n \leq 47$	3
$48 \leq n \leq 111$	4
$112 \leq n \leq 255$	5

3.4 フリーカード生成時における Batching の活用

3.3 節では，$2^{\mathsf{MBS}(n)}$-Batching が実行できるために，MS-AND プロトコルを直列で実行することを想定していた．例えば，$n = 48$ の場合，表 1 をみると $\mathsf{MBS}(48) = 4$ なので，最初の 32 個のコミットメントに対して MS-AND プロトコルを直列に実行し，フリーカードをため，残りの 2^4 個の入力に対し 2^4-Batching を適用する．しかし，このとき最初の 32 個の入力に対して，$\mathsf{MBS}(32) = 3$ より 2^3-Batching を適用できることが分かる．よって，フリーカードをためる段階でも Batching を適用することができる場合がある．

また，n の値によっては複数回 $2^{\mathsf{MBS}(n)}$-Batching を適用できることがある．例えば，$n = 36$ のとき，$\mathsf{MBS}(36) = 3$ であるので，$36 = 12 + 3 \cdot 2^3$ とかけ，2^3-Batching を 3 回適用できる．

3.5 提案プロトコル

ここでは提案プロトコルを記述する．$n \geq 2$ なる n に対して提案プロトコル $\mathcal{P}^n$ を次のように再帰的に記載する．

プロトコル $\mathcal{P}^n$

(1) $\mathsf{MBS}(n) \leq 2$ のとき，MS-AND プロトコルを $n-1$ 回実行して終了する．

(2) $\mathsf{MBS}(n) \geq 3$ のとき，$2^{\mathsf{MBS}(n)}$-Batching が j 回適用可能であるとする．

- (a) $\mathcal{P}^{n-j2^{\mathsf{MBS}(n)}}$ を実行し，フリーカード生成のためのシャッフルを 1 回行う．
- (b) $j \geq 2$ のとき，$j2^{\mathsf{MBS}(n)}$-Batching を行い，その出力に対して，$2^{\mathsf{MBS}(n)}$-Batching を実行する．
- (c) $j = 1$ のとき，$2^{\mathsf{MBS}(n)}$-Batching を実行する．
- (d) 2a と 2c の結果を入力として MS-AND プロトコルを 1 回実行して終了する．

表 2　既存研究と提案プロトコルのシャッフル回数の比較

入力数 n	既存研究	提案プロトコル
20	19	16
28	27	18
36	35	20
44	43	22
48	47	28
64	63	30
80	79	32
96	95	34
112	111	39

3.6　提案プロトコルのシャッフル回数

ここではプロトコルに必要なシャッフル回数について述べる．提案した n 入力 AND プロトコルのシャッフル回数を次に示す．n 入力 AND プロトコルのシャッフル回数の合計を s_n とすると，次のように再帰的に書ける．

$$s_n = \begin{cases} n-1 & (\mathsf{MBS}(n) \leq 2) \\ s_{n-j2^{\mathsf{MBS}(n)}} + \mathsf{MBS}(n) + 2j & (\mathsf{MBS}(n) \geq 3) \end{cases}$$

4.　既存研究との比較

本節では，既存研究 [4] と提案プロトコルのシャッフル回数の比較を表 2 と図 2 に示す．これらをみると，提案プロトコルによりシャッフル回数が削減されていることがわかる．

5.　おわりに

本稿では 2 枚の追加カードのみを用いるコミット型多入力 AND プロトコルに Batching 技術 [5] を適用することで，シャッフル回数が削減できることを示した．

謝辞　本研究は JSPS 科研費 JP19J21153，JP21K11881 の助成を受けたものです．

参考文献

[1] Ishikawa, R., Chida, E. and Mizuki, T.: Efficient Card-Based Protocols for Generating a Hidden Random Permutation Without Fixed Points, *Unconventional Computation and Natural Computation* (Calude, C. S. and Dinneen, M. J., eds.), LNCS, Vol. 9252, Cham, Springer, pp. 215–226 (online), available from ⟨https://doi.org/10.1007/978-3-319-21819-9_16⟩ (2015).

[2] Mizuki, T. and Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine, *Int. J. Inf. Secur.*, Vol. 13, No. 1, pp. 15–23 (online), available from ⟨https://doi.org/10.1007/s10207-013-0219-4⟩ (2014).

[3] Mizuki, T. and Shizuya, H.: Computational Model of Card-Based Cryptographic Protocols and Its Applications, *IEICE Trans. Fundamentals*, Vol. E100.A, No. 1, pp. 3–11 (online), available from ⟨https://doi.org/10.1587/transfun.E100.A.3⟩ (2017).

[4] Mizuki, T. and Sone, H.: Six-Card Secure AND and Four-Card Secure XOR, *Frontiers in Algorithmics* (Deng, X., Hopcroft, J. E. and Xue, J., eds.), LNCS, Vol. 5598, Berlin, Heidelberg, Springer, pp. 358–369 (online), available from ⟨https://doi.org/10.1007/978-3-642-02270-8_36⟩ (2009).

[5] Shinagawa, K. and Nuida, K.: A single shuffle is enough for secure card-based computation of any Boolean circuit, *Discrete Appl. Math.*, Vol. 289, pp. 248–261 (online), available from ⟨https://doi.org/10.1016/j.dam.2020.10.013⟩ (2021).

[6] Ueda, I., Miyahara, D., Nishimura, A., Hayashi, Y., Mizuki, T. and Sone, H.: Secure implementations of a random bisection cut, *Int. J. Inf. Secur.*, Vol. 19, No. 4, pp. 445–452 (online), available from ⟨https://doi.org/10.1007/s10207-019-00463-w⟩ (2020).

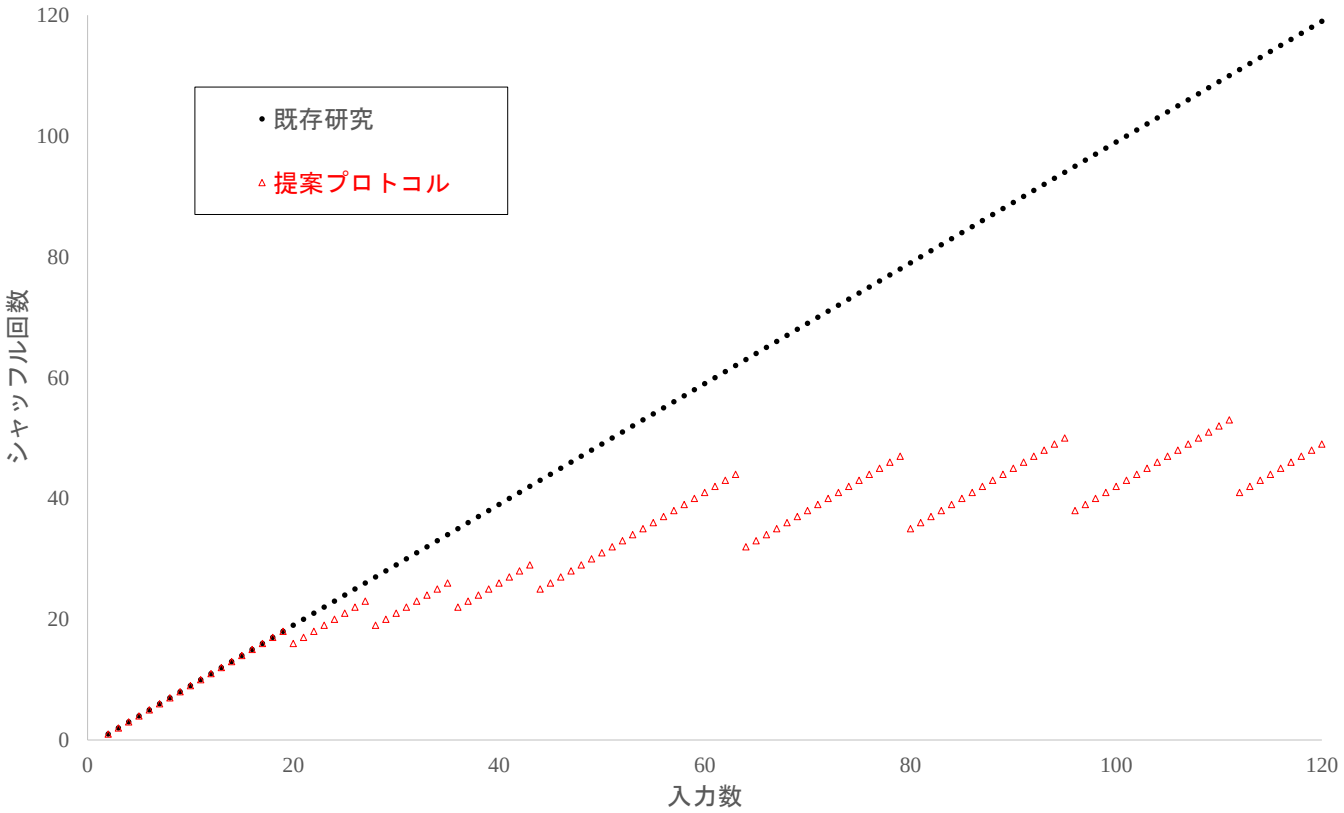


図 2　既存研究と提案プロトコルのシャッフル回数の比較