任意の関数を計算するPrivate PEZプロトコルの改善手法

安部 芳紀[1,a)]　岩本 貢[1,b)]　太田 和夫[1,c)]

概要：Balogh らは，PEZ と呼ばれるお菓子を用いてマルチパーティ計算を実現する，private PEZ プロトコルを提案した．Private PEZ プロトコルでは，初期文字列で表現されるキャンディの列を dispenser に入れる必要があり，初期文字列の長さが短いほど効率的なプロトコルとなる．Balogh らが提案した，n 入力の任意の関数を計算する private PEZ プロトコルの構成方法では，S 関数と呼ばれる関数が重要な役割を担っているが，Balogh らの論文には S 関数の導出方法に関する説明が無いため，S 関数を用いた構成方法を直感的に理解することは難しい．本研究では，直感的に理解しやすい T 関数を導入し，n 入力の任意の関数を計算する private PEZ プロトコルを構成する．T 関数は private PEZ プロトコルの再帰的な構造から導かれ，S 関数と等価な関数であることを示すことができる．さらに，S 関数の構成方法は入力数 n のみに依存するが，T 関数を用いた構成方法の再帰構造に着目することで，n 入力の任意の関数に関して有効となる private PEZ プロトコルの効率化の技法を提案する．この技法を用いることで，例えば，3 入力 XOR を計算するために必要な初期文字列の長さを，72 から 50 まで減らすことができる．

キーワード：マルチパーティ計算，Private PEZ プロトコル

How to Improve the Private PEZ Protocol for General Functions

Yoshiki Abe[1,a)]　Mitsugu Iwamoto[1,b)]　Kazuo Ohta[1,c)]

Abstract: A private PEZ protocol (PPP), proposed by Balogh et al., enables secure multi-party computation with a (big) PEZ dispenser. In PPP, a sequence of candies, called an initial string, is filled in the (big) PEZ dispenser. Therefore, the shorter string is better. In the original construction of PPP for general functions, a mysterious function called S function is used without explanation of how to derive the S function, which makes it difficult to intuitively understand the original construction. In this study, we construct PPP for general functions by introducing an intuitively clear function called T function. The T function is derived from the recursive structure of PPP and is equivalent to the S function. Furthermore, efficient PPPs are constructed from our construction. For instance, the length of the initial string used for computing 3-input XOR function gets short from 72 to 50.

Keywords: Multi-party computation, Private PEZ protocol

1. はじめに

1.1 本研究の背景

PEZ と呼ばれるお菓子を用いてマルチパーティ計算を実現するプロトコル（private PEZ プロトコル）が提案されている [1]．Private PEZ プロトコルでは，初期文字列と呼ばれる文字列で表現されるキャンディの列を dispenser に入れた後，各プレーヤが，自分の入力値に対応する，あらかじめ決められた個数のキャンディを取り出す．その結果，dispenser の先頭に残った 1 つのキャンディが表現する値が計算結果となる．各プレーヤが取り出すキャンディは，他のプレーヤの入力値に依らず，自分の入力値のみで決まるようにプロトコルが構成されているため，マルチパーティ計算で必要とされる入力値の秘匿性を保証するこ

[1] 電気通信大学
The University of Electro-Communications
[a)] yoshiki@uec.ac.jp
[b)] mitsugu@uec.ac.jp
[c)] kazuo.ohta@uec.ac.jp

とができる．Private PEZ プロトコルの効率性を測る指標の一つが初期文字列の長さであり，初期文字列の長さが短いほど効率的なプロトコルであると言える．

Balogh らは，任意の関数を計算する private PEZ プロトコルの構成方法を提案している [1]．その構成方法では，S 関数と呼ばれる関数が重要な役割を担っている．しかし，S 関数は 2 進数表現を用いて天下り的に定義されているだけで，S 関数の導出方法の関する説明が無い．そのため，S 関数を用いた構成方法は，アルゴリズムとして private PEZ プロトコルを構成することは数学的に証明できるが，なぜそのプロトコルでマルチパーティ計算が正しく実行できるのかを直感的に理解することが難しい．その結果，その構成方法を応用して，より効率的な private PEZ プロトコルを構成することも困難である．

一方，計算する関数を限定することで，効率的なプロトコルを構成することは可能である．任意 4 入力の関数を計算する場合は，6941 個のキャンディが必要である [1] が，文献 [2] では，計算する関数を 4 入力多数決に限定し，再帰的な構造を利用することで，必要なキャンディの個数を 29 個まで減らしたプロトコルを提案している．また，任意の関数を計算するプロトコルの初期文字列の長さのオーダーは $O(2^n!)$ であるが，文献 [3] では，計算する関数を対称関数に限定することで，初期文字列の長さのオーダーを $O(n \times n!)$ とするプロトコルを提案している．

1.2 本研究の概要

本研究では，T 関数と呼ばれる関数を導入し，任意の関数を計算する private PEZ プロトコルを構成する．T 関数は private PEZ プロトコルの再帰的な構造から自然に導かれ，S 関数よりも直観的な理解が容易である．また，S 関数の性質を利用すると，T 関数が S 関数と等価な関数であることを示すことができる．さらに，T 関数を用いた private PEZ プロトコルの構成方法を応用することで，計算する関数毎に効率化された private PEZ プロトコルを構成することができる．

1.3 本論文の構成

本論文の構成は次のようになっている．第 2 節では，private PEZ プロトコルの定義について述べる．第 3 節では，提案手法について述べる．第 4 節では，T 関数と S 関数の等価性について述べる．第 5 節では，本論文のまとめについて述べる．

1.4 記法

- $[a] := \{1, 2, \ldots, a\}$.
- $a \circ b$: 2 つの文字列 a, b の連結．文字列の連結であることが明らかな場合は $\circ$ を省略する．
- $\biguplus_{j:C(x)} T(j)$: 条件 $C(x)$ を満たす全ての j に関して，j の降順で文字列 $T(j)$ を連結したもの．
- $a \preccurlyeq b$: 文字列 a が文字列 b の接頭部になっている．
- $|a|$: 文字列 a の長さ
- 自然数 x の 2 進数表記を $(x_n x_{n-1} \cdots x_0)_2$ と書く．
- $x = (x_n x_{n-1} \cdots x_0)_2$ のとき，簡単の為に $f(x_n, x_{n-1}, \ldots, x_0)$ を $f(x)$ と書く．

2. Private PEZ プロトコル

PEZ とは，dispenser と呼ばれる筒の中にキャンディの列を詰めた後，筒の中のキャンディを上から 1 つずつ取り出して食べるお菓子のことである．本論文において使用する dispenser は次の 3 つの仮定を満たすものとする．

- 任意の数のキャンディを入れることができる [*1]．
- あるプレーヤは，取り出すキャンディの個数が他のプレーヤには分からないように，キャンディを取り出すことができる．
- dispenser の中に残っているキャンディの個数を知ることができない [*2]．

複数のプレーヤ $P_1, P_2, \ldots, P_n$ が，各自の入力値 $x_i \in \Sigma$ を秘匿したまま，関数 $f : \Sigma^n \to \Gamma$ の出力値 $f(x_n, x_{n-1}, \ldots, x_1)$ のみを得ることを考える．Private PEZ プロトコルとは，キャンディと dispenser を用いてマルチパーティ計算を実現するものであり，次の 2 つの要求を満たす．

正確性 (Correctness) 任意の入力値 $x = (x_n, x_{n-1}, \ldots, x_1)$ に対して，プロトコルが正確な出力値 $f(x)$ を出力する．

プライバシー 他のプレーヤの入力値に関する情報を，自分が取り出したキャンディから知ることができない．

まず，プライバシーを考慮しない（入力値を必ずしも秘匿できない）PEZ プロトコルについて説明する．

定義 1 (PEZ プロトコルの定義 [1]) n 人のプレーヤ $P_1, P_2, \ldots, P_n$ で関数 $f : \Sigma^n \to \Gamma$ を計算する PEZ プロトコルは，初期文字列 $\alpha \in \Gamma^*$ と m 個の連続した行動 $(M_1, M_2, \ldots, M_m)$ で定義される．ただし，各行動 M_j は組 (i_j, μ_j) で構成される．i_j は行動するプレーヤの添え字 $(1 \le i_j \le n)$ を表現している．$\mu_j : \Sigma \to \{0, 1, \ldots, |\alpha| - 1\}$ は，プレーヤ P_{i_j} の入力値として取りうる値から，dispenser から読み出す文字数を決める写像を表現している．

PEZ プロトコルは，次の 3 つのステップで実行される．

初期設定 初期文字列に対応するキャンディの列を dispenser に入れる [*3]．

入力（計算） 与えられた m 個の行動を実行する．各行

[*1] 実際の dispenser の容量は 12 個である．
[*2] dispenser の重さなど，物理的な情報から残りの個数を知ることもできないものとする．
[*3] $|\Gamma|$ 種類のキャンディを，Γ の各シンボルに対応させる．

表 1　XOR_2 を計算する PEZ プロトコル
初期文字列 $\alpha = 010$

行動	プレーヤ	取り出す文字数		取り出す文字列	
		$x_i = 0$	$x_i = 1$	$x_i = 0$	$x_i = 1$
M_0	P_0	0	1	–	0
M_1	P_1	0	1	–	0/1

表 2　XOR_2 を計算する private PEZ プロトコル
初期文字列 $\alpha = 101100$

行動	プレーヤ	取り出す文字数		取り出す文字列	
		$x_i = 0$	$x_i = 1$	$x_i = 0$	$x_i = 1$
M_0	P_0	0	3	–	101
M_1	P_1	0	2	–	10
M_2	P_0	1	0	1	–

動 M_j では，P_{i_j} が $\mu_j(x_{i_j})$ 個のキャンディを秘密裏に dispenser から取り出す．

出力　最後にもう 1 つキャンディを取り出す．そのキャンディが示す値が計算結果である．

$\mathrm{XOR}_2(x_1, x_0) := x_1 \oplus x_0$ を計算する PEZ プロトコルの例を，**表 1** に示す．表 1 のプロトコルの正確性は，入力値に関する 4 つ全ての場合について，実際にプロトコルを実行すれば確認できる．一方で，プライバシーは守られていない．なぜなら，表 1 の M_1 において P_1 が取り出す文字列は，M_0 で P_0 が文字列 0 を取り出す（P_0 の入力値 x_0 が 1）かどうかによって変化するためである．すなわち，P_1 は，M_1 で文字（列）0 を取り出せば $x_0 = 0$，文字（列）1 を取り出せば $x_0 = 1$ であると知ることができる．

次に，プライバシーを考慮した private PEZ プロトコルについて説明する．

定義 2 (Private PEZ プロトコル [1])　PEZ プロトコルに対し，次の 2 つの条件を満たす写像 $\beta : [m] \times \Sigma \to \Gamma^*$ が存在するとき，その PEZ プロトコルを private PEZ プロトコルと呼ぶ．

(a)　$\forall j \in [m], \forall \sigma \in \Sigma$ に対し，$|\beta(j, \sigma)| = \mu_j(\sigma)$

(b)　$\forall x \in \Sigma^n$ に対し，

$$\beta(1, x_{i_1}) \circ \beta(2, x_{i_2}) \circ \ldots \circ \beta(m, x_{i_m}) \circ f(x) \preccurlyeq \alpha$$

XOR_2 を計算する private PEZ プロトコルの例を，**表 2** に示す．入力値に関する全ての場合でプロトコルを実行すると，表 2 のプロトコルの正確性とプライバシーを確認することができる．定義 2 の 2 つの条件の直感的な意味は，各行動で取り出す文字列（view）が各プレーヤ自身の入力値のみに依存しているということであり，表 2 のように取り出す文字列の部分が他のプレーヤの入力に関わらず一意に決まるということである．一方で，プライバシーが守られない表 1 のプロトコルでは，M_1 において，$x_1 = 1$ のときに取り出す文字列が 2 通りで，一意に定まらない（すなわち，写像 β が定義できない）．

表 3　2 入力の private PEZ プロトコル
初期文字列 $\alpha_{f_2} = f(0)f(1)f(2)f(0)f(1)f(0)f(3)$

行動	プレーヤ	取り出す文字数		取り出す文字列	
		$x_i = 0$	$x_i = 1$	$x_i = 0$	$x_i = 1$
M_0	P_0	0	3	–	$f(0)f(1)f(2)$
M_1	P_1	0	2	–	$f(0)f(1)$
M_2	P_0	0	1	–	$f(0)$

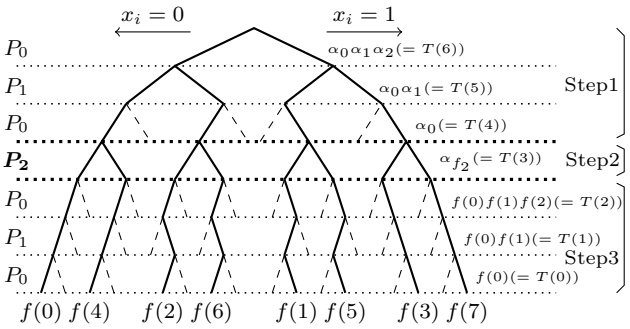

図 1　3 入力 private PEZ プロトコルを表現した 2 分木

3. 提案手法

本節では，入力値をバイナリに限定した，任意の関数を計算する private PEZ プロトコルについて述べる．任意の関数の入力値はバイナリで表現できるため，任意の関数を計算する private PEZ プロトコルを構成するには，入力値をバイナリに限定した任意の関数を計算する private PEZ プロトコルを構成できれば十分である [1]．

本研究では，n 入力の任意の関数を計算する private PEZ プロトコルが，再帰的に構成できることを示す．

3.1　2 入力から 3 入力へのプロトコルの拡張

まず，例として，2 入力のプロトコルから 3 入力のプロトコルを構成する方法を述べる．基となる 2 入力の private PEZ プロトコルを**表 3** に示す [*4]．ここでは 2 入力の private PEZ プロトコルの存在を仮定して，3 入力の privte PEZ プロトコルを構成する．**図 1** は，構成する 3 入力 private PEZ プロトコルを 2 分木で表現したものである．破線の辺は，親のノードで x_i の値が既に決まっていて，そちらの方向には進めないことを表している．

3.1.1　Step1：3 入力プロトコルの全体構造の決定

始めに，3 入力の関数を，2 つの入力に注目して以下のように 4 つの場合に分ける．

$$f(x_2, x_1, x_0) = \begin{cases} f(x_2, 0, 0) & \text{if } x_1 = 0, x_0 = 0 \\ f(x_2, 0, 1) & \text{if } x_1 = 0, x_0 = 1 \\ f(x_2, 1, 0) & \text{if } x_1 = 1, x_0 = 0 \\ f(x_2, 1, 1) & \text{if } x_1 = 1, x_0 = 1 \end{cases} \quad (1)$$

[*4] 表 3 の 2 入力のプロトコルは 1 入力のプロトコルから，1 入力のプロトコルは 0 入力のプロトコルから再帰的に構成できる．

表 4　$f(i)$ を α_i で置き換えた 2 入力の private PEZ プロトコル
初期文字列 $\alpha_{f_3} = \alpha_0\alpha_1\alpha_2\alpha_0\alpha_1\alpha_0\alpha_3$

行動	プレーヤ	取り出す文字列	
		$x_i = 0$	$x_i = 1$
M_0	P_0	–	$\alpha_0\alpha_1\alpha_2$
M_1	P_1	–	$\alpha_0\alpha_1$
M_2	P_0	–	α_0

式 (1) の 4 つの各場合において計算に使用する文字列をそれぞれ $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ とおく（例えば，$(x_1x_0)_2 = 3$ のとき，α_3 で $f(x_2, 1, 1)$ を計算すると仮定する）．α_i を用いると，3 入力のプロトコルは次のように構成できる．

(a) 2 つの入力値 $i = (x_1, x_0)_2$ に対応する行動を行い，α_i が dispenser の先頭に残るようにする．（図 1 における step1 に対応する．）

(b) 固定された 2 つの入力値 x_1, x_0 と残りの入力値 x_2 を入力として，(a) の動作後 dispenser の先頭に残っている文字列 α_i を用いて $f(x_2, x_1, x_0)$ の計算を行う．

(a) は 3 入力プロトコル全体の正確性を満たすための準備である．(a) の部分は，2 入力の private PEZ プロトコルにおいて，各 $f(i)$ を α_i にそれぞれ置き換えて実行すればプライバシーを保ちつつ実現することができる．$f(i)$ を α_i で置き換えたプロトコルを**表 4** に示す．(a) の後，(b) の部分については後で議論し，ここでは考えない．

(a) の実現　2 入力のプロトコルの正確性（$i = (x_1x_0)$ のとき $f(i)$ が dispenser の先頭に残っていること）と，$f(i)$ を α_i で置き換えることより，$i = (x_1x_0)$ のとき α_i が dispenser の先頭に残っている．

(a) のプライバシー　2 入力のプロトコルのプライバシー（P_0, P_1 の view が $f(i)$ を用いて一意に表現されている）と，α_i と $f(i)$ の置き換えも一意な操作であることから，(a) における P_0, P_1 の view が一意に定まる．

ここまでで，α_i の具体的な文字列は定まっていないが，3 入力プロトコルの初期文字列 α_{f_3} は，$\alpha_{f_3} = \alpha_0\alpha_1\alpha_2\alpha_0\alpha_1\alpha_0\alpha_3$ という構造をしていることが決まる．

3.1.2　Step2：α_i の構造の決定

(b) の部分では，式 (1) の 4 つの各場合について，残りの入力 x_2 に注目して，2 つの場合に分ける．例えば $x_1 = 0, x_0 = 0$ の場合は，

$$f(x_2, 0, 0) = \begin{cases} f(0,0,0) & \text{if } x_2 = 0 \\ f(1,0,0) & \text{if } x_2 = 1 \end{cases} \qquad (2)$$

に分け，この 2 つの場合について α_0 を用いて計算を行う．$\alpha_0 = \gamma_0 \circ \gamma_0'$ としておき，$x_2 = 0$ のときは α_0 の前半（文字列 γ_0）で，$x_2 = 1$ のときは α_0 の後半（文字列 γ_0'）で計算を行うとする．$\alpha_0 = \gamma_0 \circ \gamma_0'$ なので，$x_2 = 0$ なら P_2 は何も取り出さず，$x_2 = 1$ なら P_2 は γ_0 を取り出すという行動を追加することで，式 (2) の 2 つの場合について正確性（$x_2 = 0$ なら γ_0 が，$x_2 = 1$ なら γ_0' が dispenser の先頭に残っていること）を保証する．（追加した行動は，図 1 における step2 に対応している．）

表 5　構成途中の 3 入力の private PEZ プロトコル (1)
初期文字列 $\alpha_{f_3} = \alpha_0\alpha_1\alpha_2\alpha_0\alpha_1\alpha_0\alpha_3$

行動	プレーヤ	取り出す文字列	
		$x_i = 0$	$x_i = 1$
M_0	P_0	–	$\alpha_0\alpha_1\alpha_2$
M_1	P_1	–	$\alpha_0\alpha_1$
M_2	P_0	–	α_0
M_3	P_2	–	$\gamma_0/\gamma_1/\gamma_2/\gamma_3$

他の 3 つの場合も同様に，$x_2 = 0$ のときは α_i の前半（γ_i）で，$x_2 = 1$ のときは α_i の後半（γ_i'）で計算を行うことができるものとし，$\alpha_i = \gamma_i \circ \gamma_i'$ とおく．ここまでの動作をまとめると，**表 5** のようになる．

ここで，追加した行動（表 5 の M_3）のプライバシーの条件について考える．表 5 において，$x_2 = 1$ のときに P_2 が M_3 で取り出す文字列は $\gamma_i, i = 0, 1, 2, 3$ となる．全ての γ_i が同じ文字列ではないとすると，P_2 は $x_2 = 1$ のときに取り出す文字列 γ_i を区別することが可能であり，$i = (x_1x_0)_2$ の値に関する情報，すなわち x_1, x_0 に関する情報を P_2 は得ることができる．例えば，γ_3 のみが他の $\gamma_i, i = 0, 1, 2$ と異なっていた場合，P_2 は，γ_3 を取り出せば $(x_1x_0)_2 = 3$，すなわち $x_1 = x_0 = 1$ であることが分かり，$\gamma_i, i \neq 3$ を取り出せば $x_1 = x_0 = 1$ ではない，すなわち，x_1 と x_0 の少なくとも一方は値が 0 であるという情報を得ることができる．したがって，プライバシーの条件を守るためには，$\gamma_0 = \gamma_1 = \gamma_2 = \gamma_3 =: \gamma$ である必要がある．

ここまでで，γ_i, γ_i' の具体的な文字列は決まっていないが，α_{f_3} を構成する各 α_i について，$\alpha_i = \gamma_i\gamma_i' = \gamma\gamma_i'$ という構造が決まったことから，$\alpha_{f_3} = \gamma\gamma_0' \circ \gamma\gamma_1' \circ \gamma\gamma_2' \circ \gamma\gamma_0' \circ \gamma\gamma_1' \circ \gamma\gamma_0' \circ \gamma\gamma_3'$ という構造が決まったことになる．

3.1.3　Step3：α_i の具体的な構成

まず，γ_i を決定する．3.1.2 節より，式 (1) の 4 つの各場合について，$x_2 = 0$ であるとき，同一の文字列 γ を用いて $f(x_2, x_1, x_0)$ を計算をする必要がある．これは，2 入力プロトコルを再び用いることで実行可能である．つまり，2 入力の private PEZ プロトコルの初期文字列である α_{f_2} を γ として，P_0, P_1 で 2 入力のプロトコルを実行すれば良い．ここまでの動作をまとめると，**表 6** となる．なお，2 入力のプロトコルの正確性とプライバシーから，$x_2 = 0$ のときの $f(0, x_1, x_0)$ の計算の正確性と，M_4, M_5, M_6 における $x_2 = 0$ のときの P_0, P_1 のプライバシーについては保証できる．（γ_i を用いて $f(0, x_1, x_0)$ の計算を行うことは，図 1 において，step2 で左に進んだ場合の step3 に対応している．）

次に，γ_i' について考察する．表 6 の M_3 における P_2 のプライバシーについて考える．表 6 の M_4, M_5, M_6 において，$x_2 = 0$ のときと $x_2 = 1$ のときで，P_0, P_1 が別の文字

表 6　構成途中の 3 入力の private PEZ プロトコル (2)
初期文字列 $\alpha_{f_3} = \alpha_0\alpha_1\alpha_2\alpha_0\alpha_1\alpha_0\alpha_3$

行動	プレーヤ	取り出す文字列	
		$x_i = 0$	$x_i = 1$
M_0	P_0	–	$\alpha_0\alpha_1\alpha_2$
M_1	P_1	–	$\alpha_0\alpha_1$
M_2	P_0	–	α_0
M_3	P_2	–	α_{f_2}
M_4	P_0	–	$f(0)f(1)f(2)$
M_5	P_1	–	$f(0)f(1)$
M_6	P_0	–	$f(0)$

列を取り出す場合，x_2 に関する情報を P_0, P_1 は得ることができてしまう．したがって，P_2 のプライバシーを守るためには，γ_i' を使用して計算を行う場合の各プレーヤの view が，γ_i を使用して計算を行う場合の各プレーヤの view と一致する必要がある．よって，各 γ_i' の先頭には，γ_i で計算を行うときの各プレーヤの view を，2 入力のプロトコルの行動順に並べたもの（view(i)）があれば良い．そして，その後に，$x_2 = 1$ のときの出力 $f(1, x_1, x_0) = f(4 + i)$ を付け加えれば，計算の正確性を満たすことができる．ゆえに，各 γ_i' は次のようになる．ただし，λ は空文字列とする．

$$\begin{aligned}
\gamma_0' &= \text{view}(0) \circ f(4) = \lambda \circ \lambda \circ \lambda \circ f(4) \\
&= f(4) \\
\gamma_1' &= \text{view}(1) \circ f(5) = f(0)f(1)f(2) \circ \lambda \circ f(0) \circ f(5) \\
&= f(0)f(1)f(2)f(0)f(5) \\
\gamma_2' &= \text{view}(2) \circ f(6) = \lambda \circ f(0)f(1) \circ \lambda \circ f(6) \\
&= f(0)f(1)f(6) \\
\gamma_3' &= \text{view}(3) \circ f(7) \\
&= f(0)f(1)f(2) \circ f(0)f(1) \circ f(0) \circ f(7) \\
&= f(0)f(1)f(2)f(0)f(1)f(0)f(7)
\end{aligned}$$

これらの γ_i' を用いると，M_4, M_5, M_6 において，$f(4+i)$ を計算するときの P_0, P_1 の view が一意に定まるので，$x_2 = 1$ のときの P_0, P_1 に関するプライバシーについても保証できる．（γ_i' を用いて計算を行うことは，図 1 において，step2 で右に進んだ場合の step3 に対応している．）

3.1.4　まとめ

以上より，全ての γ_i, γ_i' が確定したため，各文字列 α_i も確定する．各 α_i の長さを求めると，次のようになる．

$$\begin{cases} |\gamma_i| = |\alpha_{f_2}| = 7 \\ |\gamma_0'| = 1 \\ |\gamma_1'| = 5 \\ |\gamma_2'| = 3 \\ |\gamma_3'| = 7 \end{cases} \Rightarrow \begin{cases} |\alpha_0| = |\gamma_i| + |\gamma_0'| = 8 \\ |\alpha_1| = |\gamma_i| + |\gamma_1'| = 12 \\ |\alpha_2| = |\gamma_i| + |\gamma_2'| = 10 \\ |\alpha_3| = |\gamma_i| + |\gamma_3'| = 14 \end{cases}$$

これらの値を表 6 に適用すると，**表 7** で示されている 3 入力の private PEZ プロトコルが完成する．完成した 3 入力のプロトコルについて図 1 を観察してみると，行動するプレーヤについては，P_2 の行動の前後で行動するプレーヤは，プレーヤの列をコピーしたものになっていることがわかる．すなわち，プレーヤの入力を固定すると，図の上部での辺の進み方が，図の下部にそのままコピーされていることが分かる．

表 7　3 入力の private PEZ プロトコル
初期文字列 $\alpha_{f_3} = \alpha_0\alpha_1\alpha_2\alpha_0\alpha_1\alpha_0\alpha_3$

行動	プレーヤ	取り出す文字数		取り出す文字列	
		$x_i = 0$	$x_i = 1$	$x_i = 0$	$x_i = 1$
M_0	P_0	0	30	–	$\alpha_0\alpha_1\alpha_2$
M_1	P_1	0	20	–	$\alpha_0\alpha_1$
M_2	P_0	0	8	–	α_0
M_3	P_2	0	7	–	α_{f_2}
M_4	P_0	0	3	–	$f(0)f(1)f(2)$
M_5	P_1	0	2	–	$f(0)f(1)$
M_6	P_0	0	1	–	$f(0)$

3.2　n 入力から $n+1$ 入力への拡張

3.1 節と同様にして，n 入力の private PEZ プロトコルを $n+1$ 入力のプロトコルに拡張することができる．基となる n 入力の private PEZ プロトコルが，行動するプレーヤのインデックスを表現する数列 l と，プレーヤが取り出す文字列を表現する T 関数を使って，**表 8** のように表現できると仮定する．ただし，l_j は数列 l の j 番目（初項を 0 番目とする）の項で，数列 l の長さは $2^n - 1$ とする．また，$T_n : \mathbb{N} \cup \{0\} \to \{f(i) \mid 0 \le i \le 2^n - 1\}^*$ で，$T_n(j)$ は $x_{l_j} = 1$ のときに行動 M_j で取り出される view を表し，n が明らかな場合は n を省略する．**図 2** は，構成する $n+1$ 入力 private PEZ プロトコルを 2 分木で表現したものである．

3.2.1　Step1：$n+1$ 入力プロトコルの全体構造の決定

始めに，$n+1$ 入力の関数を，n 個の入力に注目して以下のように 2^n 個の場合に分ける．

$$f(x_n, x_{n-1}, \ldots, x_0) = \begin{cases} f(x_n, 0, \ldots, 0, 0) & \text{if } x_{n-1} = 0, \ldots, x_1 = 0, x_0 = 0 \\ f(x_n, 0, \ldots, 0, 1) & \text{if } x_{n-1} = 0, \ldots, x_1 = 0, x_0 = 1 \\ \qquad\vdots & \\ f(x_n, 1, \ldots, 1, 1) & \text{if } x_{n-1} = 1, \ldots, x_1 = 1, x_0 = 1 \end{cases} \tag{3}$$

式 (3) の 2^n の各場合において $f(2^n x_n + i)$ の計算に使用する文字列をそれぞれ $\alpha_0, \alpha_1, \ldots, \alpha_{2^n-1}$ とおく． α_i を用いると，$n+1$ 入力のプロトコルは次のように構成できる．

(a)　n 個の入力値 $x_{n-1}, x_{n-2}, \ldots, x_0$ に対応する行動を行い，α_i が dispenser の先頭に残るようする．

(b)　固定された n 個の入力値 $x_{n-1}, x_{n-2}, \ldots, x_0$ と残り

表 8　n 入力の private PEZ プロトコル
初期文字列 $\alpha_{f_n} = T(2^n-1)$
$= T(2^n-2)T(2^n-3)\cdots T(0)f(2^n-1)$

行動	プレーヤ	取り出す文字数		取り出す文字列	
		$x_i=0$	$x_i=1$	$x_i=0$	$x_i=1$
M_{2^n-2}	$P_{l_{2^n-2}}$	0	$\lvert T(2^n-2)\rvert$	–	$T(2^n-2)$
M_{2^n-3}	$P_{l_{2^n-3}}$	0	$\lvert T(2^n-3)\rvert$	–	$T(2^n-3)$
⋮	⋮	⋮	⋮	⋮	⋮
M_0	P_{l_0}	0	$\lvert T(0)\rvert$	–	$T(0)$

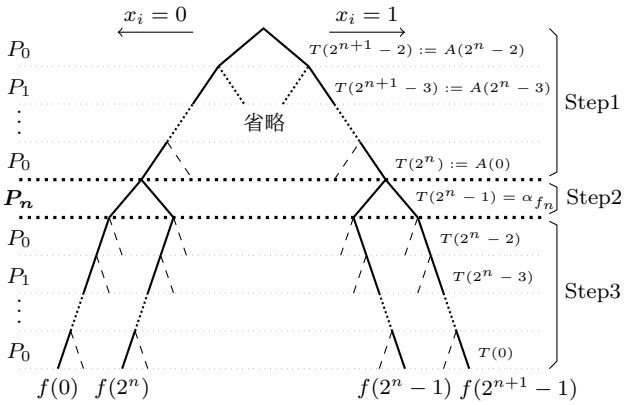

図 2　$n+1$ 入力 private PEZ プロトコルを表現した 2 分木

表 9　$f(i)$ を α_i で置き換えた n 入力の private PEZ プロトコル
初期文字列 $\alpha_{f_{n+1}} = A(2^n-1)$
$= A(2^n-2)A(2^n-3)\cdots A(0)\alpha_{2^n-1}$

行動	プレーヤ	取り出す文字列	
		$x_i=0$	$x_i=1$
$M_{2^{n+1}-2}$ [*5]	$P_{l_{2^n-2}}$	–	$A(2^n-2)$
$M_{2^{n+1}-3}$ [*5]	$P_{l_{2^n-3}}$	–	$A(2^n-3)$
⋮	⋮	⋮	⋮
M_{2^n} [*5]	P_{l_0}	–	$A(0)$

の入力値 x_n を入力として，(a) の動作後 dispenser の先頭に残っている文字列 α_i を用いて $f(x)$ の計算を行う．

(a) の部分は，n 入力の private PEZ プロトコルにおいて，各 $f(i)$ を α_i で置き換えて実行すればプライバシーを保ちつつ実現できる．任意の k, $0 \le k \le 2^n-1$ に対し，文字列 $T(k)$ の内部に含まれる $f(i)$, $0 \le i \le 2^n-1$ を α_i で置き換えた文字列を $A(k)$ とおくと，$f(i)$ を α_i で置き換えたプロトコルは**表 9** のようになる [*5]．

(a) の部分の正確性とプライバシーは，n 入力のプロトコルによって保証される．

3.2.2　Step2：α_i の構造の決定

(b) の部分では，式 (3) の 2^n 個の場合について，残り 1 つの入力 x_n に注目して，2 つの場合に分ける．例えば $x_{n-1} = x_{n-2} = \cdots = x_0 = 0$ の場合，

$$f(x_n,0,0,\ldots,0) = \begin{cases} f(0,0,0,\ldots,0) & \text{if } x_n = 0 \\ f(1,0,0,\ldots,0) & \text{if } x_n = 1 \end{cases} \quad (4)$$

のように分ける．$\alpha_i = \gamma_i \circ \gamma_i'$ としておき，$x_n = 0$ なら α_i の前半（文字列 γ_i）を，$x_n = 1$ なら α_i の後半（文字列 γ_i'）を用いて $f(x)$ の計算を行うとする．

$$\alpha_i = \gamma_i \circ \gamma_i' \quad (5)$$

式 (5) より，$x_n = 0$ なら P_n は何も取り出さず，$x_n = 1$ なら P_n は γ_i を取り出すという行動を追加すると，正確性（$f(x)$ の計算に使用する γ_i または γ_I' が dispenser の先頭に残っていること）を保証できる．ただし，3.1.2 節と同様に，追加する行動のプライバシーの条件として，$\gamma_0 = \gamma_1 = \cdots = \gamma_{2^n-1}$ である必要がある．

3.2.3　Step3：$n+1$ 入力プロトコルの具体的な構造

3.1.3 節と同様，3.2.2 節のプライバシーの条件 $\gamma_0 = \gamma_1 = \cdots = \gamma_{2^n-1}$ を満たすために，n 入力のプロトコルを再び利用する．初期文字列 α_{f_n} を γ_i として使用し，$P_0, P_1, \ldots, P_{n-1}$ で n 入力のプロトコルを実行する．

$$\gamma_i = \alpha_{f_n} = T(2^n-1) \quad (6)$$

n 入力のプロトコルの正確性とプライバシーから，$f(0, x_{n-1}, x_{n-2}, \ldots, x_0)$ の計算の正確性と，$x_n = 0$ のときの $M_{2^n-2}, M_{2^n-3}, \ldots, M_0$ における $P_0, P_1, \ldots, P_{n-1}$ のプライバシーを保証できる．

次に，γ_i' についても，3.1.3 節と同様に，$\gamma_i = \alpha_{f_n}$ を使用して n 入力の計算を行うときの各プレーヤの view を連結した後，出力 $f(1, x_{n-1}, x_{n-2}, \ldots, x_0) = f(i+2^n)$ を付け加えれば，P_n のプライバシーと，$f(1, x_{n-1}, x_{n-2}, \ldots, x_0)$ の計算の正確性と，$x_n = 1$ のときの $M_{2^n-2}, M_{2^n-3}, \ldots, M_0$ における $P_0, P_1, \ldots, P_{n-1}$ のプライバシーを保証できる．

よって，$i = (x_{n-1}x_{n-2}\ldots x_0)_2$ を入力として n 入力のプロトコルを実行した場合の，プロトコルの行動順に各プレーヤの view を並べたものを view(i) とおくと，$0 \le i \le 2^n-1$ を満たす任意の i について，γ_i' は次のように表現できる．

$$\gamma_i' = \text{view}(i) \circ f(2^n+i) \quad (7)$$

したがって，文字列 $\alpha_i = \gamma_i \circ \gamma_i'$ が確定するため，α_i の連結である $A(k)$, $0 \le k \le 2^n-1$ が確定し，$n+1$ 入力のプロトコルが完成する（**表 10**）．

さらに，表 10 のプロトコルの行動 M_{2^n+j}, $0 \le j \le 2^n-2$ において取り出される文字列 $A(j)$ で $T(2^n+j)$ を新たに定義する（$T(2^n+j) := A(j)$）．また，数列 l の長さを 2^n-1 から $2^{n+1}-1$ まで増やし，表 10 のプロトコルの行動 M_{2^n+j}, $0 \le j \le 2^n-2$ において行動するプレーヤのインデックスで l_j, $2^n-1 \le j \le 2^{n+1}-2$ を新たに定義する（$l_{2^n-1} := n, l_{2^n+k} := l_k$, $0 \le k \le 2^n-2$）．このとき，

[*5] 表 9 の行動の添え字は，後の変形を見据えて 2^n を加えた値になっている．

表 10　$n+1$ 入力の private PEZ プロトコル
初期文字列 $\alpha_{f_{n+1}} = A(2^n-1)$
$= A(2^n-2)A(2^n-3)\cdots A(0)\alpha_{2^n-1}$

行動	プレーヤ	取り出す文字数		取り出す文字列	
		$x_i=0$	$x_i=1$	$x_i=0$	$x_i=1$
$M_{2^{n+1}-2}$	$P_{l_{2^n-2}}$	0	$\lvert A(2^n-2)\rvert$	–	$A(2^n-2)$
$M_{2^{n+1}-3}$	$P_{l_{2^n-3}}$	0	$\lvert A(2^n-3)\rvert$	–	$A(2^n-3)$
⋮	⋮	⋮	⋮	⋮	⋮
M_{2^n}	P_{l_0}	0	$\lvert A(0)\rvert$	–	$A(0)$
M_{2^n-1}	$P_{l_{2^n-1}}$	0	$\lvert T(2^n-1)\rvert$	–	$T(2^n-1)$
M_{2^n-2}	$P_{l_{2^n-2}}$	0	$\lvert T(2^n-2)\rvert$	–	$T(2^n-2)$
M_{2^n-3}	$P_{l_{2^n-3}}$	0	$\lvert T(2^n-3)\rvert$	–	$T(2^n-3)$
⋮	⋮	⋮	⋮	⋮	⋮
M_0	P_{l_0}	0	$\lvert T(0)\rvert$	–	$T(0)$

表 11　T 関数表記に統一した $n+1$ 入力 private PEZ プロトコル
初期文字列 $\alpha_{f_{n+1}} = T(2^{n+1}-1)$
$= T(2^{n+1}-2)T(2^{n+1}-3)\ldots T(0)f(2^{n+1}-1)$

行動	プレーヤ	取り出す文字数	
		$x_i=0$	$x_i=1$
$M_{2^{n+1}-2}$	$P_{l_{2^{n+1}-2}}$	0	$\lvert T(2^{n+1}-2)\rvert$
$M_{2^{n+1}-3}$	$P_{l_{2^{n+1}-3}}$	0	$\lvert T(2^{n+1}-3)\rvert$
⋮	⋮	⋮	⋮
M_0	P_{l_0}	0	$\lvert T(0)\rvert$

表 10 のプロトコルは**表 11** のように表現することができる．表 11 は表 8 の n を $n+1$ に置き換えたものになっており，再帰的に $n+1$ 入力関数を計算する private PEZ プロトコルを構成していることが確認できる．

4. T 関数と S 関数の等価性

3.2 節で導入した T 関数の，形式的な定義を以下に示す．

定義 3 (T 関数)　非負の整数 x に対して，$T:\mathbb{N}\cup\{0\}\to\Gamma^*$ を次のように再帰的に定義する．

(a)　$T(0) := f(0)$[*6]

(b)　$n\geq 0$ に対し，$T(0),T(1),\ldots,T(2^n-1)$ の存在を仮定する．このとき，$0\leq k\leq 2^n-1$ に対して，

$$T(2^n+k) := A(k) \tag{8}$$

ただし，$A(k)$ は $T(k)$ 内の各シンボル $f_n(i)(0\leq i\leq k)$ を α_i に置き換えたものである．また，f_n を計算するプロトコルにおいて，入力が $i=(x_{n-1}x_{n-2}\cdots x_0)_2$ であるときの各プレーヤの view を行動順に並べたものを view(i) とすると，α_i は次のように定義される．

$$\begin{aligned}\alpha_i &:= \gamma_i\gamma_i' = \alpha_{f_n}\gamma_i'\\ &= T(2^n-1)\circ \mathrm{view}(i)\circ f_{n+1}(2^n+i)\end{aligned} \tag{9}$$

3.2 節で導入した数列 l の定義を以下に示す．

[*6] この定義は，何もせずに（何も入力しないで）$f(0)$ を出力するという 0 入力のプロトコルに対応している．

表 12　XOR_n を計算するプロトコルの初期文字列の長さの比較

n	2	3	4
T 関数（S 関数）	7	72	6941
提案手法の応用	6	50	4063
文献 [3]	7	31	165

定義 4 (数列 l)　数列 l を次のように再帰的に定義する．

(a)　初項 $l_0 := 0$

(b)　$n\geq 1$ に対し，$l_0,l_1,\ldots,l_{2^n-2}$ の存在を仮定する．このとき，次のように定義する．

$$l_{2^n-1} := n \tag{10}$$

$$l_{2^n+k} := l_k (0\leq k\leq 2^n-2) \tag{11}$$

次に，本論文の提案手法と Balogh らの手法（A.1 参照）との関係性について述べる．補題の証明については省略する．

定理 1　T 関数と S 関数は等価な関数である．

補題 1　j 非負の整数とする．数列 l の j 番目の項を l_j，j の level（[1] の Definition 12）を $l(j)$ とすると，$l_j = l(j)$ である．

補題 2 (allows（[1] の Definition 13）の解釈)　n 入力のプロトコル（表 8）において，入力が $i=(x_{n-1}x_{n-2}\cdots x_0)_2$ であるときの各プレーヤの view を行動順に並べたものを view(i) とすると，次の式が成り立つ．

$$\mathrm{view}(i) = \biguplus_{\substack{j:\ i \text{ allows } j\\ 0\leq j<i}} T(j) \tag{12}$$

補題 3　m を非負の整数，n を正の整数とする．$0\leq j<2^n-1$ のとき，m allows j は m allows (2^n+j) と同値である．

補題 4　x,j を非負の整数とする．$0\leq l(j)\leq a,b\geq 1$ のとき，x allows j は $(2^{a+b}+x)$ allows j と同値である．

証明 1 (定理 1)　任意の非負の整数 x について $T(x)=S(x)$ であることを数学的帰納法を用いて証明する．

(a)　$x=0$ のとき，$T(0)=f(0)=S(0)$ より，$T(x)=S(x)$ が成り立つ．

(b)　$x\leq k$ を満たす全ての x について $T(x)=S(x)$ が成り立つことを仮定する．ただし，$k\geq 0$ とする．このとき，

$$\begin{aligned}T(k)=S(k) &= \biguplus_{\substack{j:\ k \text{ allows } j\\ 0\leq j<k}} S(j)\circ f(k)\\ &= \biguplus_{\substack{j:\ k \text{ allows } j\\ 0\leq j<k}} T(j)\circ f(k)\end{aligned} \tag{13}$$

が成り立つ．$x=k+1$ のとき，$n=\lfloor\log_2(k+1)\rfloor$ とおくと，$k+1=2^n+m,(0\leq m\leq 2^n-1)$ と表現できる．

$$
\begin{aligned}
&T(k+1) = T(2^n + m) \\
&= A(m) && (\because \text{式 (8)}) \\
&= \biguplus_{j:\substack{0\le j<m \\ m \text{ allows } j}} A(j) \circ \alpha_m && (14) \\
&= \biguplus_{j:\substack{0\le j<m \\ m \text{ allows } j}} T(2^n + j) \circ T(2^n - 1) \\
&\quad \circ \biguplus_{j:\substack{0\le j<2^n-1 \\ m \text{ allows } j}} T(j) \circ f(2^n + m) && (\because \text{式 (8), (9), (12)}) \\
&= \biguplus_{j:\substack{2^n\le j<2^n+m \\ 2^n+m \text{ allows } j}} T(2^n + j) \circ T(2^n - 1) \\
&\quad \circ \biguplus_{j:\substack{0\le j<2^n-1 \\ 2^n+m \text{ allows } j}} T(j) \circ f(2^n + m) && (15) \\
&= \biguplus_{j:\substack{0\le j<2^n+m \\ 2^n+m \text{ allows } j}} T(j) \circ f(2^n + m) && (16) \\
&= \biguplus_{j:\substack{0\le j<k+1 \\ k+1 \text{ allows } j}} T(j) \circ f(k+1) && (\because n, m \text{ の定義}) \\
&= \biguplus_{j:\substack{0\le j<k+1 \\ k+1 \text{ allows } j}} S(j) \circ f(k+1) && (\because j \le k, \text{式 (13)}) \\
&= S(k+1). && (\because \text{(A.1)})
\end{aligned}
$$

式 (14) の等号は，$A(m)$ の定義（$T(m)$ 内の各シンボル $f(i)$ を α_i で置き換えたものであること）と，式 (13) から成り立つ．式 (15) の等号は，補題 3 の j を m に置き換え，補題 4 の l, m をそれぞれ $n-1, 1$ に置き換えたと考えれば成り立つ．式 (16) の等号は，$(2^n - 1)$ allows $(2^n - 1)$ であることから成り立つ．以上より，$x = k+1$ のときも $T(x) = S(x)$ は成り立つ．

(a), (b) より，任意の非負の整数 x について，$T(x) = S(x)$ が成り立つ． □

5. おわりに

本研究では，private PEZ プロトコルの再帰的な構造から導かれる T 関数を用いて，n 入力の任意の関数を計算するプロトコルの構成方法を示した．また，T 関数が，既存研究で使用されている S 関数と等価であることを示した．T 関数を用いた再帰的な構成方法を応用することで，n 入力の任意の関数について，関数毎に効率化されたプロトコルを構成できる．最後に，プロトコルの効率化の例として，T 関数（S 関数）を使った場合と効率化した場合の，XOR_n を計算するプロトコルの初期文字列の長さを**表 12** に示す．効率化の方針は，基にする n 入力プロトコルを効率的なプロトコルに変更して $n+1$ 入力プロトコルを構成するというものであるが，詳細は次の機会に紹介する．また，文献 [3] の方法では，対称関数限定でさらに効率化できる．

表 A·1 n 入力の private PEZ プロトコル
初期文字列 $\alpha_{f_n} = S(2^n - 1)$
$= S(2^n - 2)S(2^n - 3) \cdots S(0) f(2^n - 1)$

行動	プレーヤ	取り出す文字数		取り出す文字列	
		$x_i = 0$	$x_i = 1$	$x_i = 0$	$x_i = 1$
M_{2^n-2}	$P_{l(2^n-2)}$	0	$\lvert S(2^n-2)\rvert$	–	$S(2^n-2)$
M_{2^n-3}	$P_{l(2^n-3)}$	0	$\lvert S(2^n-3)\rvert$	–	$S(2^n-3)$
⋮	⋮	⋮	⋮	⋮	⋮
M_0	$P_{l(0)}$	0	$\lvert S(0)\rvert$	–	$S(0)$

謝辞 本研究は JSPS 科研費 JP18K19780, JP18K11293, JP18H05289, JP18H03238, JP17H01752 の助成を受けたものです．

参考文献

[1] Balogh, J., Csirik, J. A., Ishai, Y. and Kushilevitz, E.: Private computation using a PEZ dispenser, *Theoretical Computer Science*, Vol. 306, No. 1-3, pp. 69–84 (2003).
[2] 安部芳紀，山本翔太， 岩本貢，太田和夫：初期文字列が 29 文字の 4 入力多数決 Private PEZ プロトコル，電子情報通信学会技術研究報告: 信学技報，pp. 223–228 (2018).
[3] Abe, Y., Iwamoto, M. and Ohta, K.: Efficient Private PEZ Protocols for Symmetric Functions, *Theory of Cryptography Conference (TCC), to appear.*

付 録

A.1 Balogh らの構成方法 [1]

Balogh らが提案した，入力値がバイナリである任意の関数を計算できる private PEZ プロトコルの構成方法を示す．

A.1.1 構成に使用する定義

定義 5 (S 関数 ([1] の Definition 14)) 非負の整数 x に対して，$S(x)$ を次のように再帰的に定義する．

(a) $S(0) := f(0)$

(b) 任意の $x > 0$ に対し，$S(x)$ は，$j < x, x$ allows j を満たすような全ての j について，$S(j)$ を降順に連結し，最後に $f(x)$ を連結したものであり，次のように表現できる．

$$
S(x) := \biguplus_{j:\substack{0\le j<x \\ x \text{ allows } j}} S(j) \circ f(x). \qquad (A.1)
$$

A.1.2 n 入力の任意の関数を計算する private PEZ プロトコル

任意の関数 $f : \{0,1\}^n \to \Gamma$ を計算する private PEZ プロトコルは，次のように構成される．初期文字列は $S(2^n - 1)$ である．行動は $2^n - 1$ 個あり，降順で $(M_{2^n-2}, M_{2^n-3}, \ldots, M_0)$ のように実行される．各行動 M_j はプレーヤ $P_{l(j)}$ が実行し，$P_{l(j)}$ の入力値 $x_{l(j)}$ が 0 なら何も読まず（0 文字読む），入力値 $x_{l(j)}$ が 1 なら $|S(j)|$ 文字読む．このプロトコルを，**表 A·1** に示す．