

# 秘匿回路の4値論理への拡張

坂野 真聖<sup>1,a)</sup> 鈴木 幸太郎<sup>1,b)</sup>

**概要:** 2者間秘匿計算においては, Garbled Circuit を用いて計算する手法が広く用いられており, half gate 構成法などの効率的な構成法の研究が行われてきた. また, 論理の拡張の研究もおこなわれており, Garbled Circuit の3値論理への拡張が Lindell らによって提案されている. 一方 Belnap により FDE(First Degree Entailment) と呼ばれる4値論理が提案されている. FDE では True, False に加え Both と Neither が定められており, それぞれ True であり False である, True でも False でもないという状態を表している. 本研究では, Lindell らの手法の FDE と呼ばれる4値論理への拡張を提案し効率評価を行う.

**キーワード:** Garbled Circuit , 4 値論理 , 2 者間秘匿計算

## Garbled Circuit on 4-Valued Logic

SAKANO MASAKIYO<sup>1,a)</sup> SUZUKI KOUTAROU<sup>1,b)</sup>

**Abstract:** In two-party computation, the computation method using Garbled Circuit is widely used. And some efficient construct, such as freeXOR and half-gate, is widely known. In addition, there are research to extend the logic of Garbled Circuit. Concretely, Lindell et al. expanded the logic to three-valued logic. On the other hand, Belnap defined a four-valued logic called FED(First Degree Entailment). In FDE, in addition to True and False, Both and Neither are defined, Both means True and False, Neither means NOT True and NOT False. In this paper logic of Garbled Circuit expanded to FDE.

**Keywords:** Garbled Circuit , 4-Valued Logic , 2-Party computation

### 1. はじめに

#### 1.1 本研究の成果

本研究では FDE(First Degree Entailment) と呼ばれる4値論理についてガーブドサーキットで扱えるようにし, 効率化を行った. FDE の4つの状態を  $\{0, 1\}^2$  で表現し, ブール代数をあつかう(従来の)ガーブドサーキットにエンコードして効率化を行った. この効率化により, Yao の構成法による自然な拡張では16個の暗号文が必要なところを4個の暗号文で構成できる.

#### 1.2 2パーティ計算とガーブドサーキット

##### 1.2.1 2パーティ計算

2パーティ計算 [5] は, 2つの入力に対する任意の関数  $f$  を入力  $x, y$  を漏洩させずに計算することができる計算手法である. 任意の関数を計算できる万能暗号プロトコルなので, 理論的にも実用的にも重要で多くの研究がされている. 次のようなプロセスを機能性 (functionality) と呼ぶ.

$$f = (f_1, f_2), f : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^* \times \{0, 1\}^*$$

すべての入力の組み合わせ  $x, y \in \{0, 1\}^n$  に対する出力の組み合わせは, ランダム変数  $(f_1(x, y), f_2(x, y))$  となる.  $x$  を入力した1つ目のパーティは  $f_1(x, y)$  を得て,  $y$  を入力した2つ目のパーティは  $f_2(x, y)$  を得る. 機能性は頻繁に次のように定義される.

$$(x, y) \mapsto (f_1(x, y), f_2(x, y))$$

<sup>1</sup> 豊橋技術科学大学  
Toyohashi University of Technology

<sup>a)</sup> m173324@edu.tut.ac.jp

<sup>b)</sup> suzuki@cs.tut.ac.jp

2パーティ計算の安全性は記号を次のように定義する.

- $f = (f_1, f_2)$  を確率的多項式時間関数とし,  $\pi$  を  $f$  を計算する2パーティプロトコルとする
- $i$  番目のパーティの  $(x, y)$  を入力として実行しているプロトコル  $\pi$  の **view** は  $\mathbf{view}_i^\pi(x, y)$  とあらわす
- $i$  番目のパーティの  $(x, y)$  を入力として実行しているプロトコル  $\pi$  の **output** は  $\mathbf{output}_i^\pi(x, y)$  とあらわす

**定義 1.1** 確率的多項式時間アルゴリズム  $S_1, S_2$  が以下のようになる時,  $\pi$  は  $f$  を静的でセミオネストな敵 (static semi-honest adversaries) にたいして安全に計算できるといふ

$$\begin{aligned} & \{(S_1(x, f_1(x, y)), f(x, y))\}_{x, y \in \{0,1\}^*} \stackrel{c}{=} \\ & \{(\mathbf{view}_1^\pi(x, y), \mathbf{output}^\pi(x, y))\}_{x, y \in \{0,1\}^*} \\ & \{(S_2(x, f_2(x, y)), f(x, y))\}_{x, y \in \{0,1\}^*} \stackrel{c}{=} \\ & \{(\mathbf{view}_2^\pi(x, y), \mathbf{output}^\pi(x, y))\}_{x, y \in \{0,1\}^*} \end{aligned}$$

### 1.2.2 ガーブルドサーキット

ガーブルドサーキットは2パーティ計算の主な構成法で, 論理回路を暗号化して構成される. 以下では代表的な Yao の構成法 [5, 10] について示す.

$C$  を論理回路とし,  $C$  の受け取る入力を  $x, y \in \{0, 1\}^n$ , 出力を  $C(x, y) \in \{0, 1\}^n$  とする. また,  $C$  の出力ワイヤはゲート  $g$  から来ており, ゲート  $g$  のワイヤは他のゲートの入力になっていない.

$C$  中のゲート  $g$  を構成する Yao の構成法 [5, 10] を説明する.  $C$  はブール代数なので, 任意のゲート  $g$  は次の関数に置き換えられる.

$$g: \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}$$

この時の  $g$  の2つの入力ワイヤを  $w_1, w_2$  とし,  $g$  の出力ワイヤを  $w_3$  をする. さらに,  $k_1^0, k_1^1, k_2^0, k_2^1, k_3^0, k_3^1$  の6つの鍵を独立に鍵生成アルゴリズム  $G(1^n)$  により生成する. ここでは  $k_3^{g(\alpha, \beta)}$  が  $k_1^\alpha$  と  $k_2^\beta$  から計算できることが望まれる. 従って, ゲート  $g$  は以下の4つの値で構成される.

$$\begin{aligned} c_{0,0} &= E_{k_1^0}(E_{k_2^0}(k_3^{g(0,0)})) \\ c_{0,1} &= E_{k_1^0}(E_{k_2^1}(k_3^{g(0,1)})) \\ c_{1,0} &= E_{k_1^1}(E_{k_2^0}(k_3^{g(1,0)})) \\ c_{1,1} &= E_{k_1^1}(E_{k_2^1}(k_3^{g(1,1)})) \end{aligned}$$

ここで,  $E$  は chosen plaintext attacks に対して安全な共通鍵暗号化スキーム  $(G, E, D)$  である. この時のゲートの値はランダムに並べられる.

ガーブルドサーキットは表1のように様々な効率化手法が提案されており, 現在 AND ゲートは2個, XOR ゲートは0個の暗号文で構成することができる.

表1 ガーブルドサーキットの主な効率化手法  
Table 1 Efficiency method for Garbled Circuit

	AND	XOR
Yao's construction	4	4
3 row reduction [7, 8]	3	3
freeXOR [2, 4]	4	0
half gate(with freeXOR) [11]	2	0

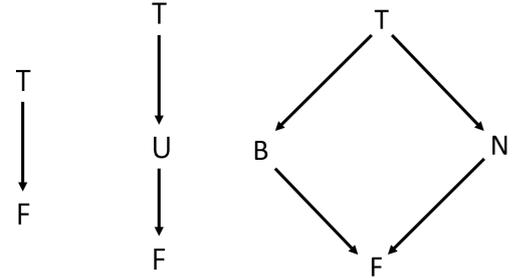


図1 2,3,4 値論理の束

Fig. 1 lattice of 2,3,4 valued logic

### 1.3 3 値論理と 4 値論理

2 値 {True, False} からなる 2 値論理, 3 値 {True, Undefined, False} からなる 3 値論理, 4 値 {True, Both, Neither, False} からなる 4 値論理を考える. 図1の左のグラフが表す半順序集合が定める束を考え, 2 値論理の AND と OR と NOT はこの束から定まる. 図1の中央のグラフが表す半順序集合が定める束を考え, Kleene の 3 値論理の AND と OR と NOT はこの束から定まる. 図1の右のグラフが表す半順序集合が定める束を考え, 4 値論理の AND と OR と NOT はこの束から決まる. Belnap の FDE と呼ばれる 4 値論理は束とは別の方法で定める NOT を持つ.

#### 1.3.1 Kleene の 3 値論理

ここで扱う 3 値論理とは, 従来のブール代数は入出力が T (Truth), F (False) の 2 値をとるのに対し, U (Undefined) を含めた T, U, F の 3 値を扱うクリーネの 3 値論理 [3] のことである.

Undefined は未定義や計算中といった意味でつかわれ, 具体的には Truth か False のどちらかではあるがどちらであるかわからない状態が Undefined である.

3 値論理における基本ゲートの動作は表2の真理値表に従う.

#### 1.3.2 Belnap の 4 値論理 (FDE)

通常のブール代数で扱う True と False に加え, Both と Neither を含めた4つの状態を扱う論理を 4 値論理といい,

表 2 3 値論理の真理値表

Table 2 Truth matrix of 3-valued logic

$\wedge_3$	T	U	F	$\oplus_3$	T	U	F	$\neg_3$	
T	T	U	F	T	F	U	T	T	F
U	U	U	F	U	U	U	U	U	U
F	F	F	F	F	T	U	F	F	T

図 1 の束から定まる 4 値論理の他に Belnap が FDE(First Degree Entailment) [1,9] と呼ばれる 4 値論理を提案している。Both は True であり False である状態を, Neither は True でも False でもない状態をそれぞれ表す。

束から定まる 4 値論理と FDE の AND( $\wedge$ ), OR( $\vee$ ), NOT( $\neg$ ) の真理値表は, それぞれ表 3, 表 4 のようになる。FDE の AND( $\wedge$ ), OR( $\vee$ ) は束から定まるが, NOT( $\neg$ ) は束から定まらない。

表 3 4 値論理の真理値表

Table 3 Truth matrix of 4-valued logic

$\wedge$	T	B	N	F	$\vee$	T	B	N	F	$\neg$	
T	T	B	N	F	T	T	T	T	T	T	F
B	B	B	F	F	B	T	B	T	B	B	N
N	N	F	N	F	N	T	T	N	N	N	B
F	F	F	F	F	F	T	B	N	F	F	T

表 4 FDE の真理値表

Table 4 Truth matrix of FDE

$\wedge$	T	B	N	F	$\vee$	T	B	N	F	$\neg$	
T	T	B	N	F	T	T	T	T	T	T	F
B	B	B	F	F	B	T	B	T	B	B	B
N	N	F	N	F	N	T	T	N	N	N	N
F	F	F	F	F	F	T	B	N	F	F	T

## 2. 先行研究

### 2.1 Lindell の手法

上で説明した 3 値論理をガブルドサーキットで扱おうとするのが, 3 値論理のガブルドサーキットである。

3 値論理は入力のパターンが増えるためテーブルも大きくなり, Yao の構成法を 3 値論理に拡張した OR ゲートの構成は表 5 のようになる。この時暗号文は 9 つ必要になる。この時, 入力ワイヤ  $w_1, w_2$ , 出力ワイヤ  $w_3$  のそれぞれの鍵は  $(k_1^X, k_2^X, k_3^X)$  ( $X = \{T, U, F\}$ ) であり,  $E_k(\cdot)$  は鍵  $k$  で  $\cdot$  を暗号化する関数である。

Lindell ら [6] により 3 値論理をブール代数に変換し, ブール代数ガブルドサーキットで計算した結果を 3 値論理に帰着させることにより 3 値論理の結果を得る方法が提案された。その方法について解説する。

3 値を表すのに 2 ビット必要なので, ここでは  $(x_T, x_F)$  で 3 値を表す。この時のブール代数と 3 値の対応関係  $R_{3 \rightarrow 2}$

表 5 Yao の構成法を 3 値論理に拡張した AND ゲート

Table 5 Yao's construction for 3-valued logic

入力ワイヤ $w_1$	入力ワイヤ $w_2$	出力ワイヤ $w_3$	garbled table
$k_1^T$	$k_2^T$	$k_3^T$	$E_{k_T}(E_{k_2^T}(k_3^T))$
$k_1^U$	$k_2^U$	$k_3^U$	$E_{k_U}(E_{k_2^U}(k_3^U))$
$k_1^F$	$k_2^F$	$k_3^F$	$E_{k_F}(E_{k_2^F}(k_3^F))$
$k_1^U$	$k_2^T$	$k_3^U$	$E_{k_U}(E_{k_2^T}(k_3^U))$
$k_1^U$	$k_2^U$	$k_3^U$	$E_{k_U}(E_{k_2^U}(k_3^U))$
$k_1^U$	$k_2^F$	$k_3^F$	$E_{k_U}(E_{k_2^F}(k_3^F))$
$k_1^F$	$k_2^T$	$k_3^F$	$E_{k_F}(E_{k_2^T}(k_3^F))$
$k_1^F$	$k_2^U$	$k_3^F$	$E_{k_F}(E_{k_2^U}(k_3^F))$
$k_1^F$	$k_2^F$	$k_3^F$	$E_{k_F}(E_{k_2^F}(k_3^F))$

は以下のようになる。

$$R_{3 \rightarrow 2} \subseteq \{T, U, F\} \times \{0, 1\}^2$$

$$R_{3 \rightarrow 2} = \{(T, (1, 1)), (F, (0, 0)), (U, (0, 1)), (U, (1, 0))\}$$

この関係  $R_{3 \rightarrow 2}$  には 3 値からブール代数に写像するとき U が 2 通りの値をとる。そのため, 2 つの容認可能な入力変換関数  $Tr_{3 \rightarrow 2}$  が存在する。どちらも T を (1,1) に F を (0,0) に写像し, 一方は U を (1,0) に写像し, もう一方は U を (0,1) に写像する。

この時の AND( $\wedge_3$ ), XOR( $\oplus_3$ ), NOT( $\neg_3$ ) は, 入力を  $(x_T, x_F)(y_T, y_F)$ , 出力を  $(z_T, z_F)$  としたときに次のように表せる。

#### AND( $\wedge_3$ )

$\wedge_3$  を入力  $(x_T, x_F, y_T, y_F)$ , 出力  $(z_T, z_F)$  で定義する。

$$z_T = x_T \wedge y_T$$

$$z_F = (x_F \wedge y_F) \oplus ((x_T \oplus x_F) \wedge (y_T \oplus y_F) \wedge (\neg(x_F \oplus y_T)))$$

この時, 4 つの AND ゲートと 4 つの XOR ゲートで構築することができる。4 つの AND ゲートは 8 つの暗号文で, 4 つの XOR ゲートは 0 個の暗号文で構築できるので, 3 値の AND ゲート  $\wedge_3$  では 8 つの暗号文が使われる。

#### XOR( $\oplus_3$ )

$\oplus_3$  を入力  $(x_T, x_F, y_T, y_F)$ , 出力  $(z_T, z_F)$  で定義する。

$$z_T = (x_T \oplus y_T) \oplus ((x_T \oplus x_F) \wedge (y_T \oplus y_F))$$

$$z_F = x_F \oplus y_F$$

この時, 1 つの AND ゲートと 5 つの XOR ゲートで構築することができる。1 つの AND ゲートは 2 つの暗号文で, 5 つの XOR ゲートは 0 個の暗号文で構築できるので, 3 値の AND ゲート  $\wedge_3$  では 2 つの暗号文が使われる。

#### NOT( $\neg_3$ )

$\neg_3$  を入力  $(x_T, x_F)$ , 出力  $(z_T, z_F)$  で定義する。

$$z_T = \neg x_T$$

$$z_F = \neg x_F$$

この時 AND ゲートも, XOR ゲートも用いないので暗号文は使われない.

このようにとると, AND(3 値) ゲートと XOR(3 値) ゲートがそれぞれ 8 個と 2 個の暗号文で表せる. これ以外にも AND(3 値) ゲートと XOR(3 値) ゲートがそれぞれ 4 個ずつで表現できる方法も提案されており, 表 6 に示す. ここでの  $aux$  は一時的に計算結果を保持している.

表 6  $xAND_4y$  における  $z$

Table 6  $z$  on  $xAND_4y$

ADN	$z_T = x_T \wedge y_T$	$z_F = x_F \wedge y_F$
XOR	$z'_T = (x_T \oplus y_T) \oplus ((x_T \oplus x_F) \wedge (y_T \oplus y_F))$	
	$z'_F = x_F \oplus y_F$	$aux = \neg z'_T \wedge z'_F$
	$z_T = z'_T \oplus aux$	$z_F = z'_F \oplus aux$
NOT	$z_T = \neg x_F$	$z_F = \neg x_T$

### 3. 提案手法

#### 3.1 構成手法

FDE に Yao の構成法を適応させると表 7 のようになり, 任意の FDE ゲートを 16 個の暗号文を使い表すことができる.

表 7 Yao の構成法を FDE に拡張した AND ゲート

Table 7 Yao's construction for FDE

入力ワイヤ $w_1$	入力ワイヤ $w_2$	出力ワイヤ $w_3$	garbled table
$k_1^T$	$k_2^T$	$k_3^T$	$E_{k_1^T}(E_{k_2^T}(k_3^T))$
$k_1^T$	$k_2^B$	$k_3^B$	$E_{k_1^T}(E_{k_2^B}(k_3^B))$
$k_1^T$	$k_2^N$	$k_3^F$	$E_{k_1^T}(E_{k_2^N}(k_3^F))$
$k_1^T$	$k_2^F$	$k_3^F$	$E_{k_1^T}(E_{k_2^F}(k_3^F))$
$k_1^B$	$k_2^T$	$k_3^B$	$E_{k_1^B}(E_{k_2^T}(k_3^B))$
$k_1^B$	$k_2^B$	$k_3^B$	$E_{k_1^B}(E_{k_2^B}(k_3^B))$
$k_1^B$	$k_2^N$	$k_3^F$	$E_{k_1^B}(E_{k_2^N}(k_3^F))$
$k_1^B$	$k_2^F$	$k_3^F$	$E_{k_1^B}(E_{k_2^F}(k_3^F))$
$k_1^N$	$k_2^T$	$k_3^N$	$E_{k_1^N}(E_{k_2^T}(k_3^N))$
$k_1^N$	$k_2^B$	$k_3^F$	$E_{k_1^N}(E_{k_2^B}(k_3^F))$
$k_1^N$	$k_2^N$	$k_3^N$	$E_{k_1^N}(E_{k_2^N}(k_3^N))$
$k_1^N$	$k_2^F$	$k_3^F$	$E_{k_1^N}(E_{k_2^F}(k_3^F))$
$k_1^F$	$k_2^T$	$k_3^F$	$E_{k_1^F}(E_{k_2^T}(k_3^F))$
$k_1^F$	$k_2^B$	$k_3^F$	$E_{k_1^F}(E_{k_2^B}(k_3^F))$
$k_1^F$	$k_2^N$	$k_3^F$	$E_{k_1^F}(E_{k_2^N}(k_3^F))$
$k_1^F$	$k_2^F$	$k_3^F$	$E_{k_1^F}(E_{k_2^F}(k_3^F))$

これをより効率的にするために FDE をブール代数ガブルドサーキットで扱う. FDE 入力を図 2 のようにブール代数に変換し計算する.

図の右側のアプローチでは変換  $Tr_{4 \rightarrow 2}$  を適用して FDE の入力をブール代数にマッピングし次に定義するブール関数を計算し最後に変換  $Tr_{2 \rightarrow 4}$  を適用して出力をマッピングし戻すことで機能します. ブール関数は変換により定義されるため, FDE 関数  $f_4$  は変換  $Tr_F$  を介して  $f_2$  に変換さ

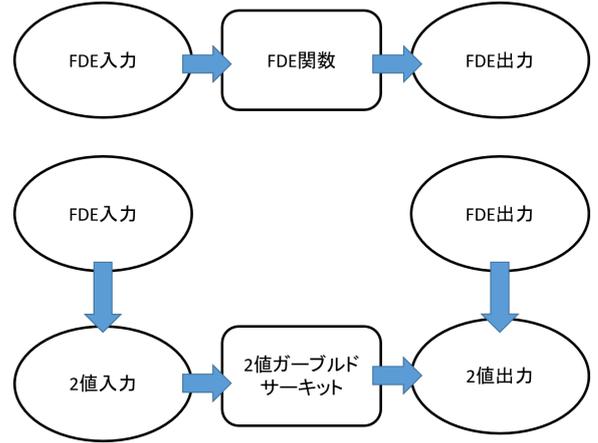


図 2 FDE をブール代数に変換するアイデア

Fig. 2 FDE encode to boolean

れる. つまり  $f_2 = Tr_F(f_4)$  であり,  $f_2$  をガブルドサーキットで計算します.

ここでは FDE の状態  $x_2$  つのブール代数の値  $(x_T, x_F)$  で表し,  $x_T$  は True であれば 1 とし  $x_F$  は False であれば 1 とする. Both は True であり False であるため  $x_T = x_F = 1$ , Nither は True でも False でもないので  $x_T = x_F = 0$  である. この値エンコードを正式に定義すると,

#### 定義 3.1

$$R_{4 \rightarrow 2} \subseteq \{T, B, N, F\} \times \{0, 1\}^2$$

$$R_{4 \rightarrow 2} = \{(T, (1, 0)), (B, (1, 1)), (N, (0, 0)), (F, (0, 1))\}$$

となる.

$m, n \in \mathbb{N}$  とする  $R_{4 \rightarrow 2}$  を値エンコードとし  $Tr_F^{n,m}$  を関数変換とします.  $(R_{4 \rightarrow 2}^m, Tr_F^{n,m})$  は, すべての  $f_3 \in F_3(m, n)$  に対して  $Tr_{4 \rightarrow 2}^{n,m}$  が有効な入力変換の場合,  $F_4(m, n)$  の FDE-ブール代数符号化である.

#### 3.1.1 AND( $\wedge$ )

FDE の状態  $x, y$  の連言 ( $AND_4(x, y)$ ) を  $z = (z_T, z_F)$  とするとき,  $z$  は表 8 の真理値表をとる.

表 8  $xAND_4y$  における  $z$

Table 8  $z$  on  $xAND_4y$

$z_T$	(1,0)	(1,1)	(0,0)	(0,1)
(1,0)	1	1	0	0
(1,1)	1	1	0	0
(0,0)	0	0	0	0
(0,1)	0	0	0	0

$z_F$	(1,0)	(1,1)	(0,0)	(0,1)
(1,0)	0	1	0	1
(1,1)	1	1	1	1
(0,0)	0	1	0	1
(0,1)	1	1	1	1

表 8 より,  $(z_T, z_F)$  は

$$z_T = x_T \wedge y_T$$

$$z_F = x_F \vee y_F$$

と定義することができる。

### 3.1.2 OR( $\vee$ )

FDE の状態  $x, y$  の選言 ( $\text{OR}_4(x, y)$ ) を  $z = (z_T, z_F)$  とするとき,  $z$  は表 9 の真理値表をとる。

表 9  $x\text{OR}_4y$  における  $z$   
Table 9  $z$  on  $x\text{OR}_4y$

$z_T$	(1,0)	(1,1)	(0,0)	(0,1)
(1,0)	1	1	1	1
(1,1)	1	1	1	1
(0,0)	1	1	0	0
(0,1)	1	1	0	0

$z_F$	(1,0)	(1,1)	(0,0)	(0,1)
(1,0)	0	0	0	0
(1,1)	0	1	0	1
(0,0)	0	0	0	0
(0,1)	0	1	0	1

表 9 より,  $(z_T, z_F)$  は

$$z_T = x_T \vee y_T$$

$$z_F = x_F \wedge y_F$$

と定義することができる。

### 3.1.3 NOT( $\neg$ )

FDE の状態  $x, y$  の否定 ( $\text{NOT}_4(x)$ ) を  $z = (z_T, z_F)$  とするとき,  $z$  は表 10 の真理値表をとる。

表 10  $\text{NOT}_4x$  における  $z$   
Table 10  $z$  on  $\text{NOT}_4x$

$x$	$z_T$	$z_F$
(1,0)	0	1
(1,1)	1	1
(0,0)	0	0
(0,1)	1	0

表 10 より,  $(z_T, z_F)$  は

$$z_T = x_F$$

$$z_F = x_T$$

と定義することができる。

ちなみに, この否定を先述の東論での NOT で定義する場合は

$$z_T = \neg x_T$$

$$z_F = \neg x_F$$

であり必要な暗号文の個数に差はない。

## 3.2 効率評価

$4 \times 4$  の真理値表を持つ回路を, Yao の構成法を用い素朴にガールドサーキットを構成すると 16 個の暗号文が必要になる。一方, 提案手法では通常のガールドサーキットで用いられる効率化手法が使えるので AND( $\wedge$ ) が 2 個, XOR( $\oplus$ ) が 0 個の暗号文で構成できる。

そのため  $\text{AND}_4$  ゲートは

$$z_T = x_T \wedge y_T$$

$$z_F = x_F \vee y_F = \neg(\neg x_F \wedge \neg y_F)$$

より, 2 個の  $\wedge$  を含むため必要な暗号文の数は 4 個

$\text{OR}_4$  ゲートは

$$z_T = x_T \vee y_T = \neg(\neg x_T \wedge \neg y_T)$$

$$z_F = x_F \wedge y_F$$

より, 2 個の  $\wedge$  を含むため必要な暗号文の数は 4 個

$\text{NOT}_4$  ゲートは

$$z_T = x_F$$

$$z_F = x_T$$

より, 0 個の  $\wedge$  を含むため必要な暗号文の数は 0 個となり, いずれのゲートにせよ Yao の構成法を用いて素朴に構成するより大幅に効率化が行えている。

## 4. ブラックボックスプロトコル

このセクションでは FDE 回路のプロトコルを説明する。既存のブール代数ガールドサーキットのプロトコルを使用し, セキュリティプロパティを直接継承できるようにする。

Lindell らの手法では, 以下の理由より 3 値をブール代数に単純にコンパイルすることが安全ではなく, 修正が必要だった。

- $\{T, U, F\}$  が 2 つ以上の値にマッピングされる可能性。
- $\{0, 1\} \times \{0, 1\}$  の範囲全体にわたって定義していない。しかし, 提案手法では  $\{T, B, N, F\}$  は 1 つの値にのみマッピングされ,  $\{0, 1\} \times \{0, 1\}$  のすべての範囲で変換が定義されているため修正の必要がなく, 以下の定理が成り立つ。

**定理 4.1**  $\pi$  をブール代数ガールドサーキットを安全に計算するためのプロトコルとし,  $f_4$  を FDE 回路  $C$  を持つ FDE 関数,  $C'$  を有効な FDE-ブール代数符号化を介して  $C$  から導出されるブール回路とする。出力変換を  $C'$  に追加した回路を  $C'_1$ , 入力変換および出力変換を  $C'$  に追加することにより得られた回路を  $C'_2$  で表す。このとき次が成り立つ。

- (1) セミオネストな攻撃者の存在下で  $\pi$  が安全な場合,  $C'_1$  でセミオネストな攻撃者の存在下でプロトコル  $\pi$  により FDE 関数  $f_4$  を安全に計算できる。

(2) 悪意のある敵の存在下で  $\pi$  が安全な場合,  $C'_2$  で悪意のある攻撃者の存在下でプロトコル  $\pi$  により FDE 関数  $f_4$  を安全に計算できる.

上記の定理は、安全な計算のためのあらゆるプロトコルに当てはまる.

## 5. おわりに

本研究では FDE をブール代数にエンコードすることによりガーブルドサーキットを FDE に適応させた. ブール代数にエンコードすることにより, すでに研究されているブール代数ガーブルドサーキットの効率化手法とブール代数ガーブルドサーキットの安全なプロトコルを継承することができる. 表 11 のように 3 値論理のガーブルドサーキットと比較しても (FDE の XOR が定義されていないため一概には言えないが) 遜色のない暗号文の数である. 3 値論理は削減を行わなかったら 9 個の暗号文で構成できるが, FDE では削減を行わなかったら 16 個の暗号文が必要であるため, 3 値論理の場合に比べて大きな効率改善となっている.

表 11 本研究と Lindell の手法の比較

Table 11 This paper and Lindell's construction

	AND	OR	XOR	NOT
本研究	4	4	-	0
Lindell の手法 (1)	8	8	2	0
Lindell の手法 (2)	4	4	4	0

謝辞 支えてくれたすべての人に感謝します.

## 参考文献

- [1] Nuel D Belnap. A useful four-valued logic. In *Modern uses of multiple-valued logic*, pp. 5–37. Springer, 1977.
- [2] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the security of the “free-xor” technique. In *Theory of Cryptography Conference*, pp. 39–53. Springer, 2012.
- [3] Stephen Cole Kleene, NG de Bruijn, J de Groot, and Adriaan Cornelis Zaanen. *Introduction to metamathematics*, Vol. 483. van Nostrand New York, 1952.
- [4] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free xor gates and applications. In *International Colloquium on Automata, Languages, and Programming*, pp. 486–498. Springer, 2008.
- [5] Yehuda Lindell and Benny Pinkas. A proof of security of Yao’s protocol for two-party computation. *Journal of Cryptology*, Vol. 22, No. 2, pp. 161–188, 2009.
- [6] Yehuda Lindell and Avishay Yanai. Fast garbling of circuits over 3-valued logic. In *IACR International Workshop on Public Key Cryptography*, pp. 620–643. Springer, 2018.
- [7] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM conference on Electronic commerce*, pp. 129–139. ACM, 1999.
- [8] Benny Pinkas, Thomas Schneider, Nigel P Smart, and

Stephen C Williams. Secure two-party computation is practical. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 250–267. Springer, 2009.

- [9] Geoff Sutcliffe, Francis Jeffrey Pelletier, and Allen Hazen. Making belnap’s “useful 4-valued logic” useful. In *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida, USA. May 21-23 2018.*, pp. 116–121, 2018.
- [10] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pp. 162–167. IEEE, 1986.
- [11] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 220–250. Springer, 2015.