秘密計算および関連する話題について

縫田 光司（NUIDA, Koji）

九州大学 マス・フォア・インダストリ研究所（IMI）
（nuida@imi.kyushu-u.ac.jp）

IMI 共同利用「若手研究者のための暗号周辺分野の
横断的・萌芽的研究の創発」@九州大学
2026 年 6 月 22 日

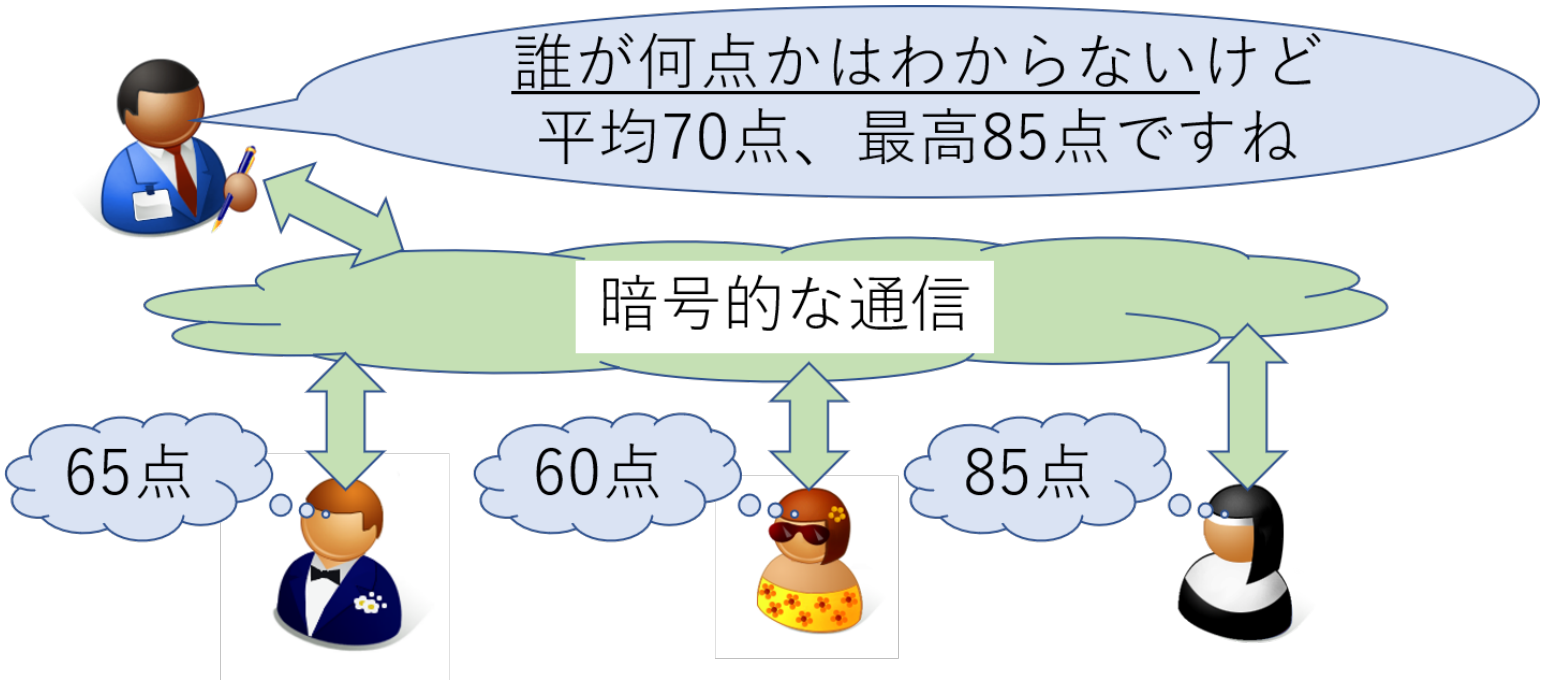
目次

- 秘密計算とは
- 主な手法１：準同型暗号
- 主な手法２：秘匿回路
- 主な手法３：秘密分散
- 特殊な秘密計算のモデル
- カードを用いた秘密計算
- 秘密計算の安全性について

目次

- 秘密計算とは
- 主な手法１：準同型暗号
- 主な手法２：秘匿回路
- 主な手法３：秘密分散
- 特殊な秘密計算のモデル
- カードを用いた秘密計算
- 秘密計算の安全性について

秘密計算の模式図



秘密計算とは

- secure multiparty computation (MPC) [Yao, 1982][1]
- 複数の人（パーティ）がそれぞれ秘密の入力を持っている状況で、互いに通信しながら下記を実現する暗号技術
  - 正当性：全員のデータに関する何らかの値を正しく計算する
  - 安全性：通信の内容から、各自の入力データが別の参加者に漏れない（安全性についての詳細は後述）
- 以下、2パーティ（P_0 と P_1）の場合を考える

[1] A. C.-C. Yao: Protocols for Secure Computations. FOCS 1982

参考文献（宣伝）

- 縫田 光司、大畑 幸矢、菊池 亮、
『暗号理論による秘密計算から
プライバシー保護データ解析へ』、
サイエンス社、2025 年 11 月

秘密計算プロトコルの主な構成要素

- 秘匿回路（garbled circuit）
- 秘密分散（secret sharing）
- 準同型暗号（homomorphic encryption）
- …
  - これらが混在したプロトコルもある
- 効率性を度外視すれば、あらゆる関数の秘密計算が可能 [GMW, 1987][1]

[1] O. Goldreich, S. Micali, A. Wigderson: How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. STOC 1987

目次

- 秘密計算とは
- 主な手法１：準同型暗号
- 主な手法２：秘匿回路
- 主な手法３：秘密分散
- 特殊な秘密計算のモデル
- カードを用いた秘密計算
- 秘密計算の安全性について

暗号化と復号

- 暗号化：暗号化鍵を用いて、平文を暗号文に変換する
- 復号：復号鍵を用いて、暗号文を平文に戻す
- 安全性（概要）：暗号文（と公開情報）があっても平文の情報がわからない
- 共通鍵型の方式：暗号化鍵と復号鍵が同一（かつ秘密）
- 公開鍵型の方式：復号鍵のみが秘密、暗号化鍵は公開可能

準同型暗号

- 以下、$[[m]]$ で平文 m の暗号文を表す
- 暗号文 $[[m]], [[m']]$ から直接（復号せず）、中身の平文の演算結果 $m * m'$ に対する暗号文 $[[m * m']]$ を得ることができる（主に公開鍵型の）暗号化
- 暗号文に対する操作：「準同型演算」
- 複数の種類の演算が可能な方式もある

例：パイエ暗号（加法準同型暗号）[Paillier, 1999][1]

- 暗号化鍵は $N = pq$（p, q は同じビット長の異なる素数）、復号鍵は $\mathrm{lcm}(p-1, q-1)$
- 平文 $m \in \mathbb{Z}/N\mathbb{Z}$ の暗号文：
  $[[m]] := (1+N)^m r^N \bmod N^2$
  （$r \in (\mathbb{Z}/N\mathbb{Z})^\times$ は乱数）
- 復号については割愛
- $c_i = (1+N)^{m_i} r_i^N$（$i = 1, 2$）から和の暗号文 $c_1 c_2 = (1+N)^{m_1+m_2}(r_1 r_2)^N$ を作れる
- スカラー倍の暗号文も作れる：
  $c^s = (1+N)^{s \cdot m}(r^s)^N$

[1] P. Paillier: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. EUROCRYPT 1999

応用例1：紛失通信

- （1-out-of-n）oblivious transfer (OT)
- P_0（送信者）の入力：$x_0, \ldots, x_{n-1}$
- P_1（受信者）の入力：$r \in \{0, \ldots, n-1\}$
- 送信者は何の情報も得ず、
  受信者は x_r のみを得る
  （他の x_i については情報を得ない）
- 多くの秘密計算プロトコルの構成要素技術

紛失通信のプロトコル

1. 受信者はパイエ暗号の暗号化鍵と復号鍵を生成し、暗号化鍵を送信者に送る
   また、暗号文 $[[\delta_0]], \ldots, [[\delta_{n-1}]]$ を送信者に送る、ただし $\delta_r := 1$、$\delta_i := 0$（$i \neq r$）
2. 送信者は準同型演算で $[[\sum_{i=0}^{n-1} x_i \cdot \delta_i]]$ を作り受信者に送る
   - $\sum_{i=0}^{n-1} x_i \cdot \delta_i = x_r$
3. 受信者は復号して x_r を得る

- 注：紛失通信についてはより（大幅に）効率的な手法がある

応用例2：秘匿一致判定

- P_0 と P_1 が入力 x_0 と x_1 をもっているとして、入力の値を互いに秘密にしたままで $x_0 = x_1$ であるかだけを判定したい
- x_0, x_1 は非負整数で、p, q よりも小さいとする（注：p, q はとても大きい）

秘匿一致判定のプロトコル

1. P_0 はパイ エ暗号の暗号化鍵 N と復号鍵を生成し、暗号化鍵を P_1 に送る
   また、暗号文 $[[-x_0]]$ を P_1 に送る
2. P_1 は暗号文 $[[x_1]]$ を作り、
   準同型演算で $[[x_1 - x_0]]$ を作る
   また、$s \in (\mathbb{Z}/N\mathbb{Z})^\times$ を一様ランダムに選び、
   準同型演算で $[[s \cdot (x_1 - x_0)]]$ を作り P_0 に送る
3. P_0 は復号し、0 が出れば「入力が一致」

秘匿一致判定のプロトコル：正当性と安全性

- P_0 は暗号化鍵と暗号文しか P_1 に送っていないので、P_0 の入力は P_1 に秘密のまま

P_0 は $[[s \cdot (x_1 - x_0)]]$ を復号し、
0 が出れば「入力が一致」

- $x_0 = x_1$ のとき：復号結果は常に 0
- $x_0 \neq x_1$ のとき：$0 < |x_1 - x_0| < p, q$ より $x_1 - x_0$ は $\mathbb{Z}/N\mathbb{Z}$ において可逆
  - よって $s \cdot (x_1 - x_0)$ は $(\mathbb{Z}/N\mathbb{Z})^\times$ の一様ランダムな元であり、特に 0 ではない
  - その分布は x_1 によらないので x_1 の情報は（「$\neq x_0$」以外）P_0 に秘密のまま

完全準同型暗号 [Gentry, 2009][1]

- あらゆる演算が可能な準同型暗号
  - 秘密計算プロトコルの構成が楽になる
- 代表的な方式（の一部）：BFV [FV, 2012][2], FHEW [DM, 2015][3], CKKS [CKKS, 2017][4], TFHE [CGGI, 2020][5]
- 既存方式はどれも、準同型演算による暗号文ノイズの増大を伴う
  - 「ノイズフリー」な構成は未解決問題

[1] C. Gentry: Fully Homomorphic Encryption Using Ideal Lattices. STOC 2009
[2] J. Fan, F. Vercauteren: Somewhat Practical Fully Homomorphic Encryption. ePrint 2012
[3] L. Ducas, D. Micciancio: FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. EUROCRYPT 2015
[4] J. H. Cheon, A. Kim, M. Kim, Y. S. Song: Homomorphic Encryption for Arithmetic of Approximate Numbers. ASIACRYPT 2017
[5] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène: TFHE: Fast Fully Homomorphic Encryption Over the Torus. J. Cryptology 2020

目次

- 秘密計算とは
- 主な手法１：準同型暗号
- 主な手法２：秘匿回路
- 主な手法３：秘密分散
- 特殊な秘密計算のモデル
- カードを用いた秘密計算
- 秘密計算の安全性について

秘匿回路 [Yao, 1986][1]：概要

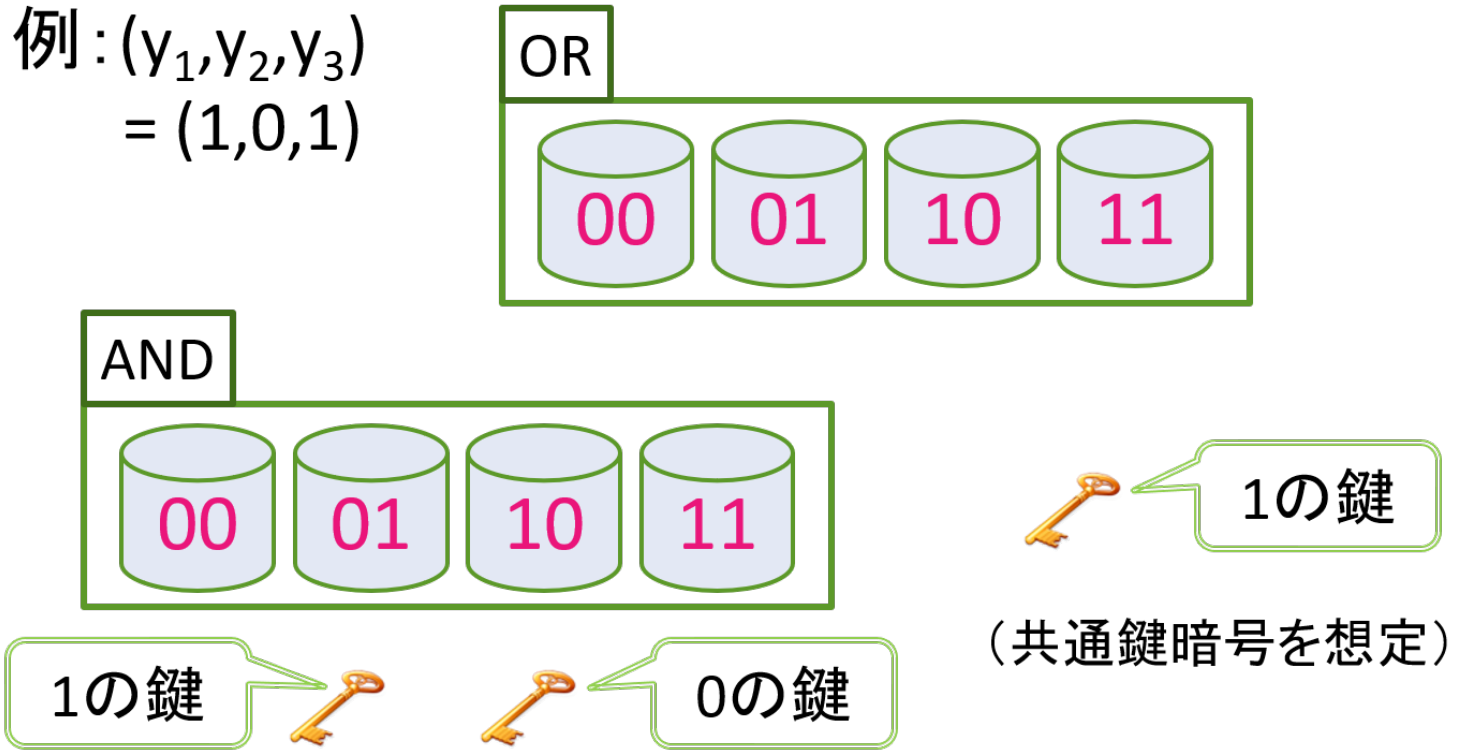
- P_0 が自分の入力を決めると、
  計算したい値は P_1 の入力のみの関数
- P_0 はこの関数を回路図に変換
- 各ゲート C を 4 個の暗号文の組に変換
  - 「鍵 0 または鍵 1」と「鍵 $0'$ または鍵 $1'$」を用いて二重に暗号化
  - 鍵 b と b' で暗号化された暗号文には、次のゲートの入力 $C(b, b')$ に対応する復号鍵が入っている
- 最初に紛失通信を用いて、P_1 の入力に対応する鍵だけを P_1 に渡す

[1] A. C.-C. Yao: How to Generate and Exchange Secrets. FOCS 1986

秘匿回路の模式図

（Bさんは $F(y) = (y_1 \text{ AND } y_2) \text{ OR } y_3$ を計算）

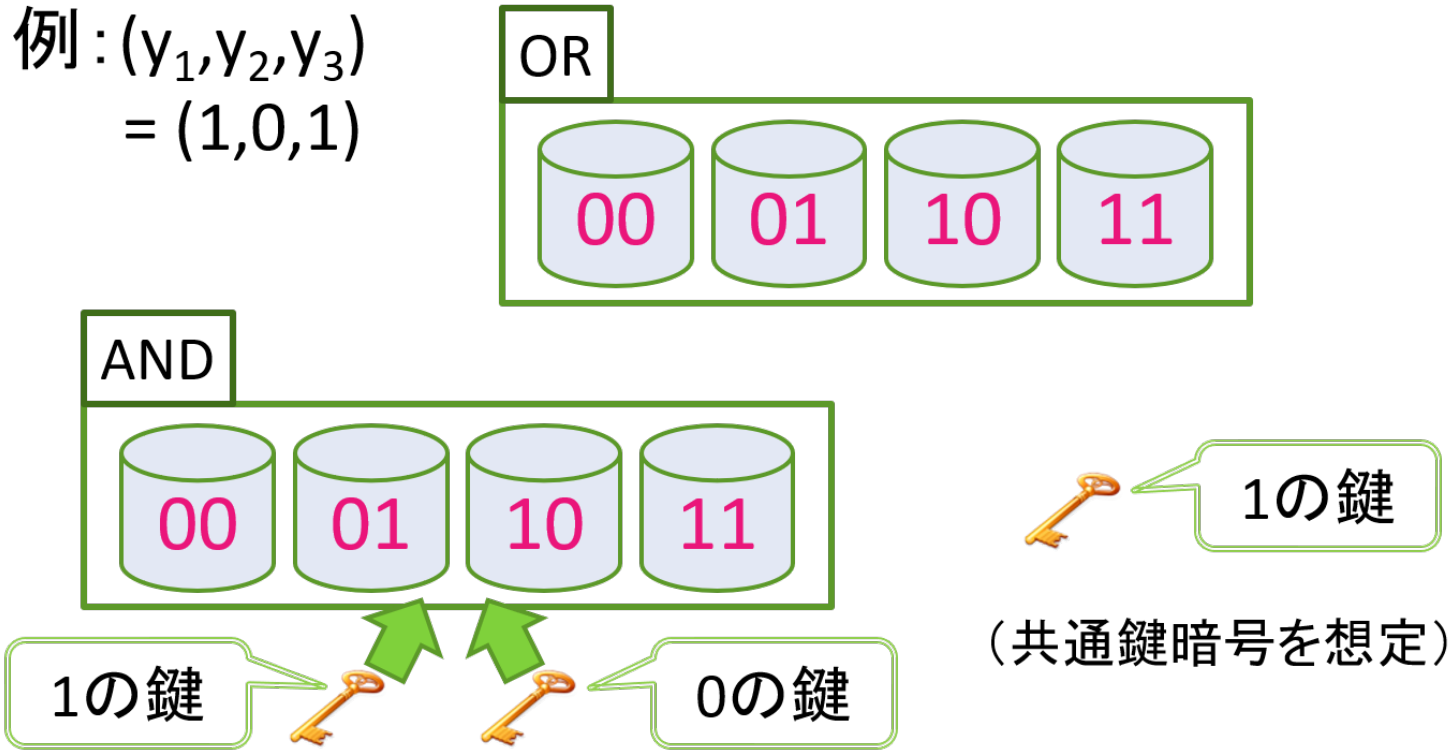
例：(y_1,y_2,y_3)
$= (1,0,1)$



秘匿回路の模式図

（Bさんは $F(y) = (y_1 \text{ AND } y_2) \text{ OR } y_3$ を計算）

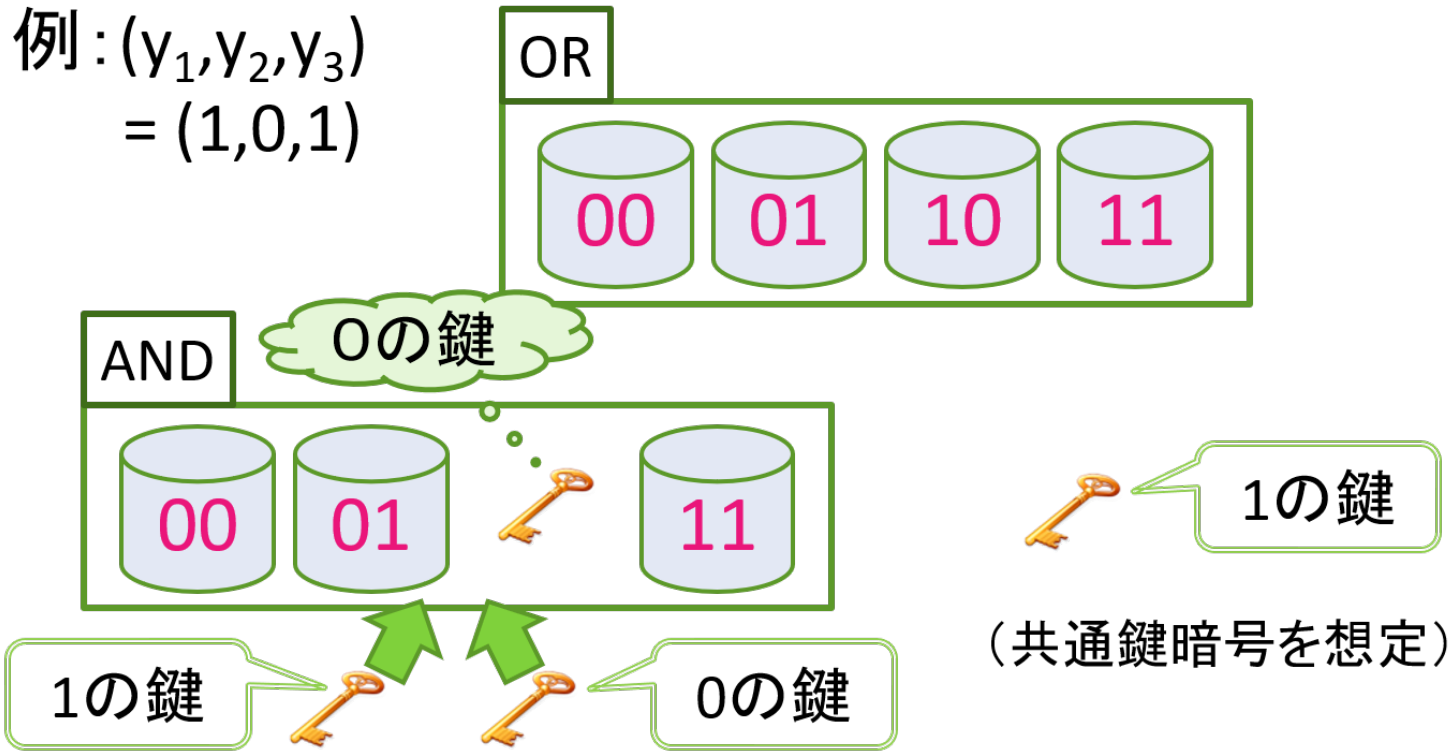
例：(y_1,y_2,y_3)
$= (1,0,1)$



秘匿回路の模式図

（Bさんは $F(y) = (y_1 \text{ AND } y_2) \text{ OR } y_3$ を計算）

例：$(y_1,y_2,y_3) = (1,0,1)$

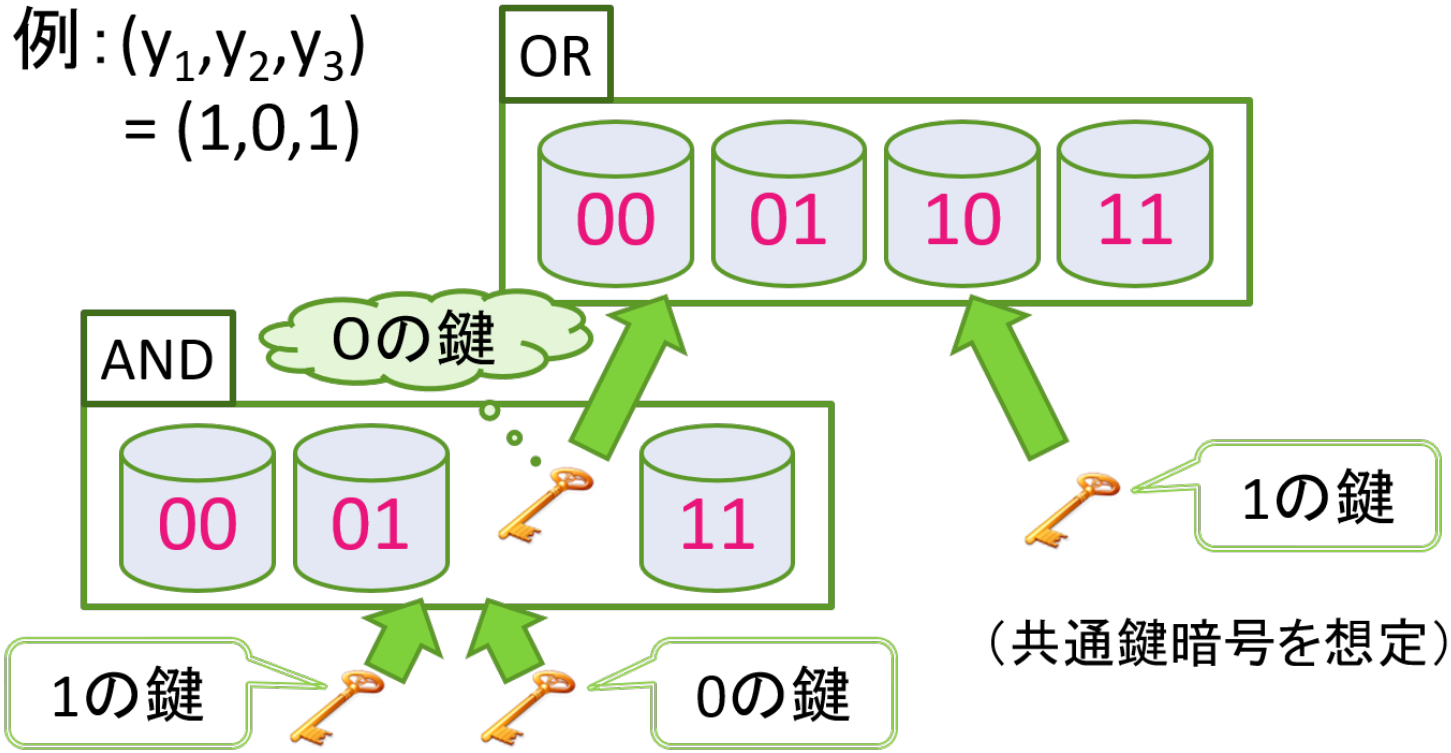


（共通鍵暗号を想定）

秘匿回路の模式図

（Bさんは $F(y) = (y_1 \text{ AND } y_2) \text{ OR } y_3$ を計算）

例：$(y_1,y_2,y_3) = (1,0,1)$

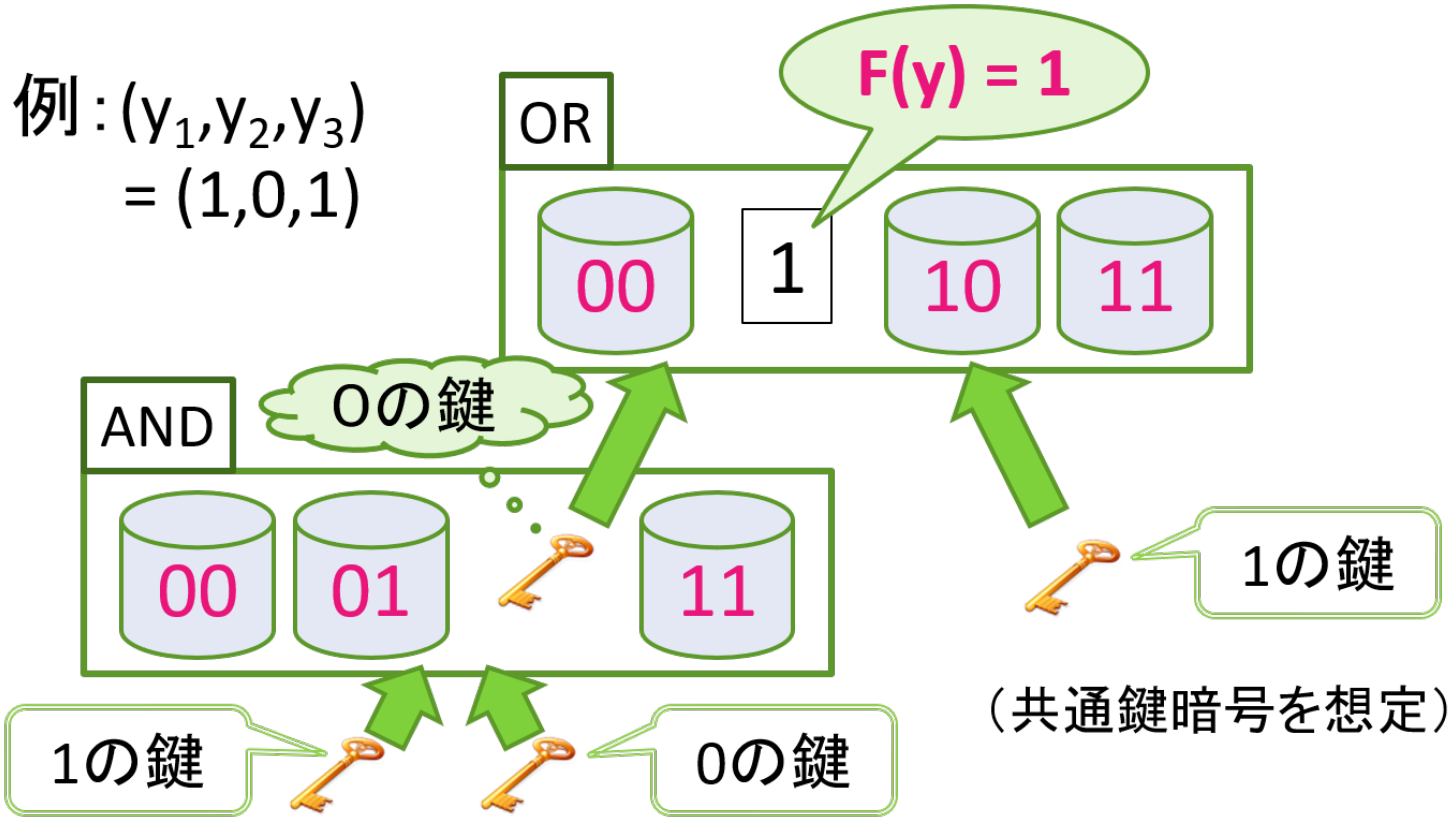


（共通鍵暗号を想定）

秘匿回路の模式図

（Bさんは $F(y) = (y_1 \text{ AND } y_2) \text{ OR } y_3$ を計算）

例：$(y_1, y_2, y_3) = (1,0,1)$



（共通鍵暗号を想定）

秘匿回路の効率化手法

- free-XOR [KS, 2008][1]
- half-gates [ZRE, 2015][2]
- BitGC [LWYY, 2025][3]
- ...

[1] V. Kolesnikov, T. Schneider: Improved Garbled Circuit: Free XOR Gates and Applications. ICALP 2008
[2] S. Zahur, M. Rosulek, D. Evans: Two Halves Make a Whole – Reducing Data Transfer in Garbled Circuits Using Half Gates. EUROCRYPT 2015
[3] H. Liu, X. Wang, K. Yang, Y. Yu: BitGC: Garbled Circuits with 1 Bit per Gate. EUROCRYPT 2025

目次

- 秘密計算とは
- 主な手法１：準同型暗号
- 主な手法２：秘匿回路
- 主な手法３：秘密分散
- 特殊な秘密計算のモデル
- カードを用いた秘密計算
- 秘密計算の安全性について

（ビットの）2-out-of-2 加法的秘密分散

- ビット x を、P_0 の持つビット（シェア）$[[x]]_0$ と P_1 の持つシェア $[[x]]_1$ の組 $[[x]] = ([[x]]_0, [[x]]_1)$ で表現、ただし $[[x]]_0 \oplus [[x]]_1 = x$
- シェアの生成：$[[x]]_0$ を一様ランダムに選び、$[[x]]_1 := x \oplus [[x]]_0$ とする
  - 片方のシェアだけからは平文 x の情報が何も得られない
- シェアの XOR をとると XOR のシェアになる：$[[x]]_i \oplus [[y]]_i = [[x \oplus y]]_i$
- ビット反転：$[[\neg x]]_0 := \neg[[x]]_0$, $[[\neg x]]_1 := [[x]]_1$
  - これらは通信なしで実行可能

乗算（AND）の秘密計算

- **Beaver Triple** (BT) [Beaver, 1992][1] :
  $c = a \wedge b$ を満たすシェア $[[a]] = ([[a]]_0, [[a]]_1)$, $[[b]] = ([[b]]_0, [[b]]_1)$, $[[c]] = ([[c]]_0, [[c]]_1)$
  - P_0 は $[[a]]_0, [[b]]_0, [[c]]_0$ のみを持ち、P_1 は $[[a]]_1, [[b]]_1, [[c]]_1$ のみを持つ
- 値 x_i に対して、例えば $x_i' := x_i \oplus [[a]]_i$ を相手に送っても x_i の情報は漏れない
- BT は例えば 1-out-of-4 OT で事前計算可能
  - P_0 の $([[a]]_1[[b]]_1)_2$ 番目の入力を $[[c]]_1$ の値に設定、P_1 が $[[c]]_1$（のみ）を得る

[1] D. Beaver: Efficient Multiparty Protocols Using Circuit Randomization. CRYPTO 1991

乗算（AND）の秘密計算

- 入力のシェア $[[x]]$ と $[[y]]$ を分配しておく

1. P_0 は $x'_0 := [[x]]_0 \oplus [[a]]_0$ と $y'_0 := [[y]]_0 \oplus [[b]]_0$ を P_1 に送り、P_1 は $x'_1 := [[x]]_1 \oplus [[a]]_1$ と $y'_1 := [[y]]_1 \oplus [[b]]_1$ を P_0 に送る
2. それぞれ $x' := x'_0 \oplus x'_1$ と $y' := y'_0 \oplus y'_1$ を計算
   - $x' = x \oplus a$, $y' = y \oplus b$
3. P_0 は $[[z]]_0$ を、P_1 は $[[z]]_1$ を計算：
   $[[z]]_0 := x' \wedge y' \oplus x' \wedge [[b]]_0 \oplus y' \wedge [[a]]_0 \oplus [[c]]_0$,
   $[[z]]_1 := x' \wedge [[b]]_1 \oplus y' \wedge [[a]]_1 \oplus [[c]]_1$

乗算（AND）の秘密計算

$$
\begin{aligned}
&[[z]]_0 \oplus [[z]]_1 \\
&= x' \wedge y' \oplus x' \wedge b \oplus y' \wedge a \oplus c \\
&= x' \wedge (y' \oplus b) \oplus (y \oplus b) \wedge a \oplus c \\
&= x' \wedge y \oplus a \wedge y \oplus a \wedge b \oplus (a \wedge b) \\
&= (x' \oplus a) \wedge y = x \wedge y
\end{aligned}
$$

- よって $[[z]] = ([[z]]_0, [[z]]_1)$ は $z = x \wedge y$ の正しいシェアとなる
- 以下、$[[z]] = [[x]] \wedge [[y]]$ と表記する

平文の大小比較の秘密計算

- 例：P_0 は整数 $x = (x_1 x_0)_2$ を、P_1 は $y = (y_1 y_0)_2$ を持ち、$x < y$ かどうかを秘密計算したい
- 以下、(Cond) は条件 Cond が真のとき 1 を、偽のとき 0 を表す
- $(x_k < y_k) = ((x_k, y_k) = (0, 1)) = (\neg x_k) \wedge y_k$ より、$[[\neg x_k]]_0 := \neg x_k$, $[[\neg x_k]]_1 := 0$, $[[y_k]]_0 := 0$, $[[y_k]]_1 := y_k$ とすると $[[(x_k < y_k)]] := [[\neg x_k]] \wedge [[y_k]]$ は $(x_k < y_k)$ の正しいシェアとなる

平文の大小比較の秘密計算

- $(x_1 = y_1) = (\neg x_1) \oplus y_1$ より、
  $[[(x_1 = y_1)]] := [[\neg x_1]] \oplus [[y_1]]$ は $(x_1 = y_1)$ の正しいシェアとなる
- $(x < y) = ((x_1 < y_1) \oplus ((x_1 = y_1) \wedge (x_0 < y_0)))$ より、
  $[[(x_1 < y_1)]] \oplus ([[(x_1 = y_1)]] \wedge [[(x_0 < y_0)]])$ は $(x < y)$ の正しいシェアとなる
- より入力の桁数が多くても同様の計算が可能

目次

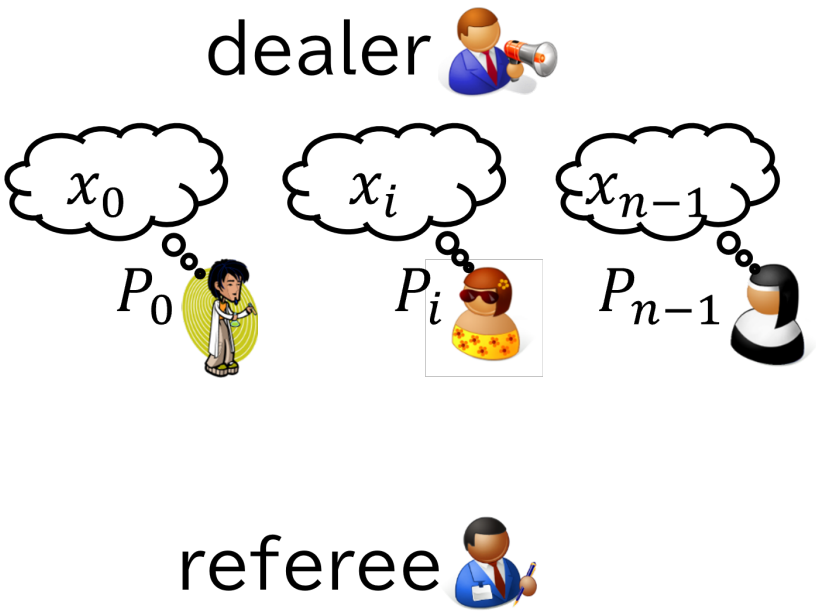
- 秘密計算とは
- 主な手法１：準同型暗号
- 主な手法２：秘匿回路
- 主な手法３：秘密分散
- 特殊な秘密計算のモデル
- カードを用いた秘密計算
- 秘密計算の安全性について

特殊な秘密計算のモデル

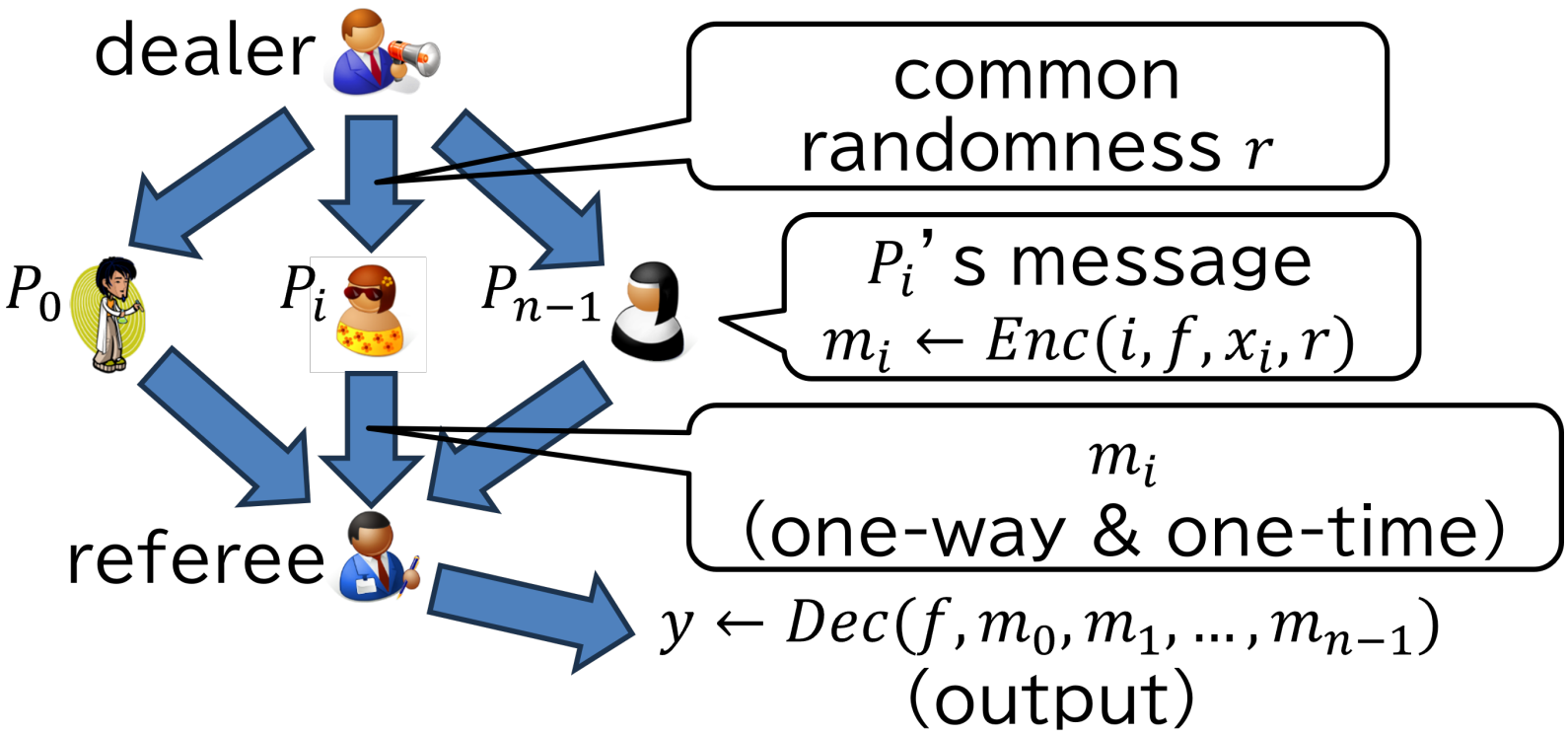
- PIR (Private Information Retrieval) [CGKS, 1995][1]
- PSM (Private Simultaneous Messages) [FKN, 1994][2]
- CDS (Conditional Disclosure of Secrets) [GIKM, 1998][3]
- ...

[1] B. Chor, O. Goldreich, E. Kushilevitz, M. Sudan: Private Information Retrieval. FOCS 1995
[2] U. Feige, J. Kilian, M. Naor: A Minimal Model for Secure Computation. STOC 1994
[3] Y. Gertner, Y. Ishai, E. Kushilevitz, T. Malkin: Protecting Data Privacy in Private Information Retrieval Schemes. STOC 1998

PSM の模式図



PSM の模式図


dealer
common
randomness r
P_0
P_i
P_{n-1}
P_i's message
$m_i \leftarrow Enc(i, f, x_i, r)$
m_i
(one-way & one-time)
referee
$y \leftarrow Dec(f, m_0, m_1, \ldots, m_{n-1})$
(output)

PSM のモデル

- 入力 x_i を持つパーティ P_i（$i = 0, \ldots, n-1$）
- 入力を持たない dealer と referee
  - dealer は入力パーティに共通の乱数（common randomness, CR）を配る
- パーティ P_i はメッセージ m_i を referee に送る
- referee は $(m_0, \ldots, m_{n-1})$ から
  プロトコルの出力 y を計算
- 正当性：$y = f(x_0, \ldots, x_{n-1})$
- 安全性：$(m_0, \ldots, m_{n-1})$ からは入力の情報が（y の値以外）漏れない

平方剰余を用いたPSMプロトコルの例

- $n = 2$, $f(x_0, x_1) = x_0 \wedge x_1$ $(x_i \in \{0, 1\})$
- CR: $(\mathbb{F}_{11})^\times$ の一様ランダムな平方剰余（平方数）r および $s_0 + s_1 \equiv 0 \pmod{11}$ を満たす一様ランダムな組 $(s_0, s_1) \in (\mathbb{F}_{11})^2$
- メッセージ：

  $m_0 := r(3 + x_0) + s_0$, $m_1 := r(2x_1) + s_1$
- 出力：$m_0 + m_1$ が $(\mathbb{F}_{11})^\times$ の平方剰余のとき $y := 0$、そうでないとき $y := 1$
  - $m_0 + m_1 = r(3 + x_0 + 2x_1)$

平方剰余を用いたPSMプロトコルの例

- 平方剰余 r 倍を無視すると、

$$m_0 + m_1 = 3 + x_0 + 2x_1 = \begin{cases} 3 & (x_0, x_1) = (0,0) \\ 4 & (x_0, x_1) = (1,0) \\ 5 & (x_0, x_1) = (0,1) \\ 6 & (x_0, x_1) = (1,1) \end{cases}$$

- $(\mathbb{F}_{11})^\times$ で $3, 4, 5$ は平方剰余、6 は平方非剰余
- よって $y = 1 \Leftrightarrow (x_0, x_1) = (1,1) \Leftrightarrow x_0 \wedge x_1 = 1$
- 同様に、どの関数も、充分大きな素数 p について $(\mathbb{F}_p)^\times$ の平方（非）剰余の列に「埋め込めて」PSMプロトコルを構成できる [SESN, 2023][1]

[1] K. Shinagawa, R. Eriguchi, S. Satake, K. Nuida: Private Simultaneous Messages Based on Quadratic Residues. Designs, Codes and Cryptography 2023

目次

- 秘密計算とは
- 主な手法１：準同型暗号
- 主な手法２：秘匿回路
- 主な手法３：秘密分散
- 特殊な秘密計算のモデル
- カードを用いた秘密計算
- 秘密計算の安全性について

カードベース暗号

- 電子的なコンピュータではなく、
  トランプカードのような物理的な道具で
  秘密計算を行う分野
- あらゆる関数の秘密計算が可能
- Five-Card Trick [Boer, 1990][1]：
  最初のカードプロトコル（$x_0 \wedge x_1$ の計算）

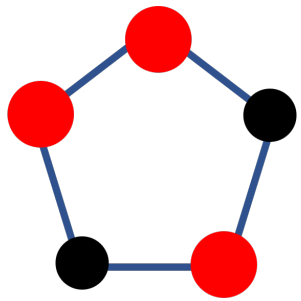
[1] B. den Boer: More Efficient Match-Making and Satisfiability: The Five Card Trick. EUROCRYPT 1989

Five-Card Trick

- （説明の都合で表現を変えている）

1. 正五角形のルーレットを用意
2. 中央（一番上）の頂点に赤カードを伏せて置く
3. 各自は自分の側の２頂点に、
   入力が 1 なら「上に赤、下に黒」、
   入力が 0 なら「上に黒、下に赤」、
   の並びでカードを伏せて置く

例



Five-Card Trick

1. 正五角形のルーレットを用意
2. 中央（一番上）の頂点に赤カードを伏せて置く
3. 各自は自分の側の 2 頂点に、
   入力が 1 なら「上に赤、下に黒」、
   入力が 0 なら「上に黒、下に赤」、
   の並びでカードを伏せて置く
4. ルーレットを充分に回してからカードを
   すべて開ける（注：シャッフルの代わり）
5. 赤が 3 枚続いたら 1 を出力、そうでなければ
   0 を出力

- 安全性：$(1,1)$ 以外の入力に対するカードの
  パターンは回転に関して合同

カードベース暗号と通常の秘密計算の関係

- カードプロトコルによる秘匿回路の実現 [SN, 2021][1]
- カードプロトコルからPSMプロトコルへの変換法 [SN, 2025][2]
- PSMプロトコルからカードプロトコルへの変換法 [ES, 2025][3]

[1] K. Shinagawa, K. Nuida: A Single Shuffle Is Enough for Secure Card-Based Computation of Any Boolean Circuit. Discrete Applied Mathematics 2021
[2] K. Shinagawa, K. Nuida: Card-Based Protocols Imply PSM Protocols. STACS 2025
[3] R. Eriguchi, K. Shinagawa: Single-Shuffle Full-Open Card-Based Protocols for Any Function. arXiv 2025

目次

- 秘密計算とは
- 主な手法１：準同型暗号
- 主な手法２：秘匿回路
- 主な手法３：秘密分散
- 特殊な秘密計算のモデル
- カードを用いた秘密計算
- 秘密計算の安全性について

秘密計算の安全性モデル

- **semi-honest モデル**：どのパーティも
  プロトコルの仕様を正しく守る
  （その上で、プロトコル中に受信した
  データから相手の入力を推測する）
- **malicious モデル**：一部のパーティは
  プロトコルの仕様を守らない可能性がある
  - malicious モデルでの安全性の方が強い
    （が、実現しにくい）
- 以降では 2 パーティ（P_0 と P_1）の
  semi-honest モデルを考える
  - 3 パーティ以上の場合は結託も考慮

安全性定義の考え方

- 例えばパーティ P_0 について、プロトコル中に P_0 が得る全データを P_0 の **view** と呼ぶ：
  - P_0 の入力
  - P_0 が用いる乱数
  - P_0 が相手から受信したデータ（の列）
- P_0 の入力と出力から、P_0 の view の分布を（多項式時間で）模倣できるとき、プロトコルは P_0 に対して安全と定める
  - 「view からは入出力以外の情報が何も得られない」

準備：分布の識別

- セキュリティパラメータ λ の関数 $\varepsilon(\lambda) \geq 0$ が無視できる（negligible）
  $\overset{\text{def}}{\Leftrightarrow} \forall k \in \mathbb{Z}_{>0}\ \exists \lambda_0\ \forall \lambda > \lambda_0\ (\varepsilon(\lambda) < 1/\lambda^k)$
  - 直感的説明：「λ が充分大きい範囲では、どの多項式の逆数よりも小さい」
- 分布の列 $(X_\lambda)_\lambda, (Y_\lambda)_\lambda$ に対する識別者 D の優位度（advantage）
  $:= |\Pr[D(X_\lambda) = 1] - \Pr[D(Y_\lambda) = 1]|$
  - X_λ の値または Y_λ の値がランダムに渡されたとき、どちらの値だったかを、$1/2$ よりもどの程度高確率で当てられるか

安全性定義 [Goldreich, 2004][1]

- $x = (x_0, x_1)$：入力、(f_0, f_1)：計算する関数
- 以下を満たす多項式時間シミュレータ S が存在するとき、プロトコルは P_0 に対して安全と定義する：
  - 組 $(S(x_0, f_0(x)), f_0(x), f_1(x))$ の分布と組 $(\mathrm{View}_0(x), \mathrm{Out}_0(x), \mathrm{Out}_1(x))$ の分布が、どの多項式時間（非一様）識別者でも無視できる優位度でしか識別できない
  - ただし $\mathrm{View}_0(x)$ は P_0 の view、$\mathrm{Out}_i(x)$ はプロトコルにおける P_i の出力を表す

[1] O. Goldreich: Foundations of Cryptography II: Basic Applications. Cambridge University Press 2004

安全性定義についての話者の研究

$$(S(x_0, f_0(x)), f_0(x), f_1(x))$$
$$\overset{c}{\approx} (\mathsf{View}_0(x), \mathsf{Out}_0(x), \mathsf{Out}_1(x))$$

- 安全性定義における識別者に、相手側の出力 $f_1(x)$ や $\mathsf{Out}_1(x)$ を渡すことの妥当性の考察 [N, 2026][1]

[1] K. Nuida: On Definitions for Semi-Honest Secure Multiparty Computation. IEICE Trans. Fundamentals 2026

安全性定義についての話者の研究

$$(S(x_0, f_0(x)), f_0(x), f_1(x))$$
$$\overset{c}{\approx} (\mathsf{View}_0(x), \mathsf{Out}_0(x), \mathsf{Out}_1(x))$$

- 安全なプロトコルの乱数を暗号学的疑似乱数（PRG）に置き換えた際に安全性が失われる具体例の提示 [N, 2021][1]
  - 直感的説明：view に P_0 の乱数が含まれるため、PRG の種が識別者に見えてしまい、PRG の安全性が役に立たない
- 安全性が保たれる充分条件 [HN, 2022][2]

[1] K. Nuida: Cryptographic Pseudorandom Generators Can Make Cryptosystems Problematic. PKC 2021
[2] N. Heseri, K. Nuida: Computational Irrelevancy: Bridging the Gap between Pseudo- and Real Randomness in MPC Protocols. IWSEC 2022

目次

- 秘密計算とは
- 主な手法１：準同型暗号
- 主な手法２：秘匿回路
- 主な手法３：秘密分散
- 特殊な秘密計算のモデル
- カードを用いた秘密計算
- 秘密計算の安全性について