コンテンツセキュリティ　　2020 年 6 月 15 日

秘匿計算 2

執筆者: 安永憲司

1　秘密分散と紛失通信による実現

次に，秘密分散を使って秘匿計算を実現する方法を紹介する．二者間での計算について説明するが，3 人以上への拡張も可能である．

P_1 が入力 $x \in \{0,1\}^n$，P_2 が入力 $y \in \{0,1\}^n$ をもち，$f(x,y)$ を秘匿計算したい．ここでも，f を計算するための論理回路を活用する．実現のアイディアは，回路の各ワイヤの値を二つのシェアに秘密分散し，P_1, P_2 はそれぞれ分散したシェアのまま手元で計算していくことである．ワイヤ i の値が $w_i \in \{0,1\}$ のとき，ランダムに $a_i \in \{0,1\}$ を選び，$w_i = a_i \oplus b_i$ とすれば，w_i をシェア a_i と b_i に分散できる．これをすべての入力ワイヤに対して行う．そして，P_1 が片方のシェア $\{a_i\}$ を，P_2 が残りのシェア $\{b_i\}$ をもち，ワイヤの値でなく，シェアの値で論理回路を計算する．すべてのワイヤ i について，$w_i = a_i \oplus b_i$ という関係を保ったまま計算を行う．最後に回路の出力ワイヤのシェアを出し合えば，出力結果を復元できる．

はじめに，P_1 は入力 $x = (x_1, \ldots, x_n)$ の各ビット $x_i \in \{0,1\}$ に対し，ランダムに $a_x^i \in \{0,1\}$ を選び，$b_x^i = x_i \oplus a_x^i$ とし，$b_x^1, \ldots, b_x^n$ を P_2 へ送る．同様に，P_2 も入力 y の各ビット $y_i \in \{0,1\}$ に対しランダムに $b_y^i \in \{0,1\}$ を選び，$a_y^i = y_i \oplus b_y^i$ とし，$a_y^1, \ldots, a_y^n$ を P_1 へ送る．回路の各入力ワイヤ i に対し，P_1 は a_i, P_2 は b_i をもち，ワイヤの値は $w_i = a_i \oplus b_i$ という関係を保っている．ランダムにビットを選べば，$\{a_i\}, \{b_i\}$ の一方のみからは，入力 x, y に関する情報はもれない．

次に，素子の演算である．論理回路は，XOR 素子，NOT 素子，AND 素子で構成されるとしよう．XOR 素子は，そのまま手元で XOR を計算すればよい．ワイヤ i, j の値がそれぞれ w_i, w_j のとき，$w_i \oplus w_j = (a_i \oplus b_i) \oplus (a_j \oplus b_j) = (a_i \oplus a_j) \oplus (b_i \oplus b_j)$ が成り立つからである．NOT 素子については，$w_i \oplus 1 = (a_i \oplus 1) \oplus b_i$ であるため，P_1 の方だけが NOT をすれば実現できる．

残るのが AND 素子である．入力ワイヤが i, j，出力ワイヤが k のとき，$w_k = w_i \wedge w_j = (a_i \oplus b_i) \wedge (a_j \oplus b_j)$ の計算のため，$w_k = a_k \oplus b_k$ を満たす a_k と b_k を P_1 と P_2 の手元に残したい．P_1 が $a_k \in \{0,1\}$ をランダムに選んだとき，$b_k \in \{0,1\}$ は次の表のように計算できる．

b_i	b_j	$w_k = (a_i \oplus b_i) \wedge (a_j \oplus b_j)$	b_k
0	0	$m_{00} = a_i \wedge a_j$	$m_{00} \oplus a_k$
0	1	$m_{01} = a_i \wedge \neg a_j$	$m_{01} \oplus a_k$
1	0	$m_{10} = \neg a_i \wedge a_j$	$m_{10} \oplus a_k$
1	1	$m_{11} = \neg a_i \wedge \neg a_j$	$m_{11} \oplus a_k$

つまり，P_2 には，(b_i, b_j) の値により伝えるべき b_k の値が変わる．ここで，紛失通信を利用する．目隠し回路の際は，入力が 2 つあり，そのうち 1 つを受けとるプロトコル Π_{OT} を使ったが，ここでは，入力が 4 つあり，そのうち 1 つを受けとるような紛失通信を使えばよい．このような $(1,4)$-紛失通信は，前回資料にある $(1,2)$-紛失通信プロトコルを使って実現できる．図 1 がそのプロトコルである．暗号化にランダムオラクルを用いているが，暗号化として必要な性質は，目隠し関数表のものと同じであり，擬似ランダム関数や二重暗号化を使って実現してもよい．

すべての素子の演算が終わると，P_1, P_2 はそれぞれ回路の出力値のシェアをもっている．そのシェアの値を相手に教えることで，出力を得ることができる．図 2 は，$(w_1 \oplus w_2) \wedge w_3$ を計算する場合の，P_1 と P_2 それぞれが手元で行なうワイヤの計算を表している．

プロトコル $\Pi_{\mathsf{OT}}^{(1,4)} = (S, R)$

送り手 S は入力 $x_0, x_1, x_2, x_3 \in \{0,1\}^\ell$，受け手 R は入力 $y \in \{0,1,2,3\}$ をもつ．

1. 送り手は，鍵 $k_0^0, k_1^0, k_0^1, k_1^1 \in \{0,1\}^n$ をランダムに選び，各 $z \in \{0,1,2,3\}$ に対し，ランダムオラクル H を使って，暗号文 $c_z = H(k_{z_0}^0, k_{z_1}^1) \oplus x_z$ をつくり，R へ送る．ここで，$(z_0, z_1) \in \{0,1\}^2$ は z のビット表現である．
2. 各 $i \in \{0,1\}$ に対し，S が入力 (k_0^i, k_1^i) の送り手，R が入力 y_i の受け手として，$(1,2)$-紛失通信を実行する．ここで，$(y_0, y_1) \in \{0,1\}^2$ は y のビット表現である．
3. R は，$k_{y_0}^0$ と $k_{y_1}^1$ を受けとり，復号結果 $c_y \oplus H(k_{y_0}^0, k_{y_1}^1)$ を出力する．

図 1　$(1,2)$-紛失通信にもとづく $(1,4)$-紛失通信

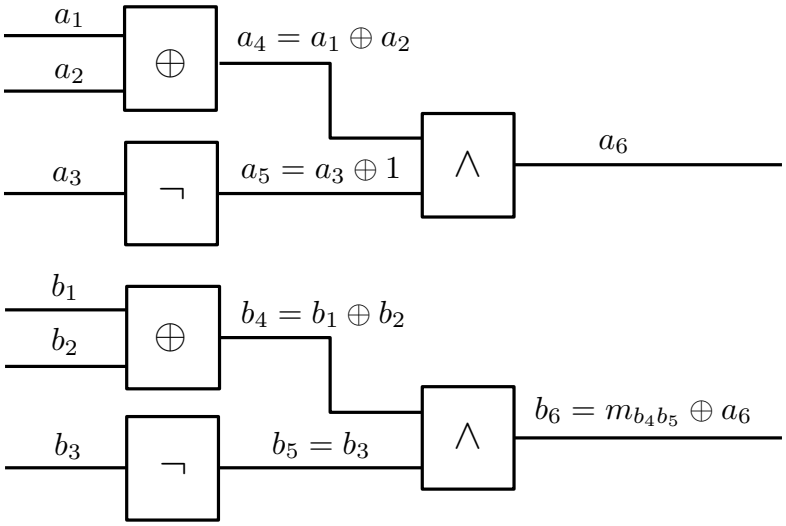

図 2　秘密分散と紛失通信による秘匿計算

3 人以上への拡張

参加者が $P_1, P_2, \ldots, P_n$ の n 人で，それぞれ入力 $x_1, x_2, \ldots, x_n$ をもち，$f(x_1, x_2, \ldots, x_n)$ を計算したい場合を考える．2 人の場合と同様に，回路の各ワイヤの値を n 個のシェアに秘密分散し，各 P_u は，分散されたシェアのまま手元で計算を行う．参加者 P_u の入力値に対応するワイヤ i の値 w_i については，P_u が $w_i = s_i^1 \oplus \cdots \oplus s_i^n$ を満たすように，$s_i^1, \ldots, s_i^n \in \{0,1\}$ をランダムに選び，P_v にシェア s_i^v を渡せばよい．実際には，自分以外のシェア $\{s_i^v\}_{v \neq u}$ をランダムに選び，自身のシェアを $s_i^u = w_i \oplus \left(\bigoplus_{v \neq u} s_i^v\right)$ と計算すればよい．

手元で行う素子の演算について，XOR 素子はそのまま手元で計算すればよく，NOT 素子は，P_1 だけが演算すればよい．残る AND 素子について，入力ワイヤ i, j，出力ワイヤ k のとき，

$$\begin{aligned} w_k = w_i \wedge w_j &= (s_i^1 \oplus \cdots \oplus s_i^n) \wedge (s_j^1 \oplus \cdots \oplus s_j^n) \\ &= \left(s_i^1 s_j^1 \oplus \cdots \oplus s_i^n s_j^n\right) \oplus \bigoplus_{u \neq v} (s_i^u \wedge s_j^v) \end{aligned}$$

と表すことができる．このうち，$s_i^u s_j^u$ は各 P_u が手元で計算できる．そして，$(s_i^u \wedge s_j^v) \oplus (s_i^v \wedge s_j^v)$ は，P_u と P_v の間で計算する．2 人の場合の AND 演算と同様に，P_v のもつ (s_i^v, s_j^v) の値より 4 通りあり，P_u がランダムに $\mu_u \in \{0,1\}$ を選び，$\mu_v = (s_i^u \wedge s_j^v) \oplus (s_i^v \wedge s_j^u) \oplus \mu_u$ を 4 通り用意し，$(1,4)$-紛失通信により μ_v を P_v に伝えればよい．

ここで用いた $w_i = s_i^1 \oplus \cdots \oplus s_i^n$ という秘密分散は，$n-1$ 個の $\{s_i^j\}$ がわかっても w_i の値は特定されないため，

$n-1$ 人の参加者が結託してもワイヤの情報は復元されない．つまり，n 人のうち $n-1$ 人が（準正直な）敵対者として結託したとしても，残った 1 人の入力に関する情報はもれない．

2 秘密分散による実現

紛失通信は，通常の設定では計算量的な安全性しか達成できない．しかし，前節のように紛失通信を使わずに秘匿計算を実現する方法がある．秘密分散方式にもとづいており，情報理論的な安全性を達成する．つまり，敵対者の計算能力を制限しなくても安全性が保証される．

Shamir の秘密分散方式を思い出そう．有限体 $\mathbb{F}$ の要素である秘密 $s \in \mathbb{F}$ に対し，$\mathbb{F}$ 上の次数 t 以下の多項式 $p(x)$ として $p(0) = s$ を満たすものをランダムに選ぶ．異なる元 $\alpha_1, \ldots, \alpha_n \in \mathbb{F} \setminus \{0\}$ に対し，$p(\alpha_i)$ を i 番目のシェアとする方式である．この場合，任意の t 個のシェアを集めても，多項式 p は定まらず，秘密 s に関する情報はもれない．以降では，秘密 s に対し次数 t 以下の多項式 p でつくられるシェアの列を

$$[s; p]_t = (p(\alpha_1), \ldots, p(\alpha_n))$$

と表すことにする．多項式 p を明示しないとき，単に $[s]_t$ と書く．

これまでに紹介した方法と同様に，秘匿計算したい関数 f を計算する回路が与えられるとする．今回は，ワイヤが 0 と 1 の二値をとる論理回路ではなく，ワイヤが有限体 $\mathbb{F}$ の元を値としてとり，素子が加算 $+$ と乗算 $\times$ で構成される**算術回路** *(arithmetic circuit)* を考える．有限体に含まれる $0, 1 \in \mathbb{F}$ を使い，ワイヤの値 $x, y \in \{0, 1\}$ に対し，$1 - x$ で NOT 素子，$x \times y$ で AND 素子を実現できるため，論理回路で計算できる関数は，算術回路でも計算できる．ここで，$1 - x$ は，1 の加法逆元 -1 を使い，$1 + (-1) \times x$ と計算すればよい．

各参加者 P_u は，算術回路の入力ワイヤで，自身の入力に対応するワイヤ i の値 w_i について，秘密分散により $[w_i, p]_t = (p(\alpha_1), \ldots, p(\alpha_n))$ というシェアをつくり，他の参加者 P_v に，シェア $p(\alpha_v)$ を渡す．すべての入力ワイヤに対応するシェアを受けとった後，各参加者は手元で算術回路をシェアの状態で計算していく．加算素子 $(+)$ のためには，$[a; p_a]_t$ と $[b; p_b]_t$ から，$[a + b; p_c]_t$ を各 $u = 1, \ldots, n$ が手元でつくれればよい．これは，Shamir の秘密分散の加法性より簡単に実現できる．各 P_u は，$p_c(\alpha_u) = p_a(\alpha_u) + p_b(\alpha_u)$ と計算すればよい．多項式 $p_c(x) = p_a(x) + p_b(x)$ は次数が t 以下であり，$p_c(0) = p_a(0) + p_b(0)$ を満たしている．

残るは乗算素子 $(\times)$ である．秘密 s に定数 $\alpha \in \mathbb{F}$ を掛けるのであれば，$[\alpha s; \alpha p]_t = (\alpha p(\alpha_1), \ldots, \alpha p(\alpha_n))$ が成り立つため，それぞれ手元で α を掛ければよい．次に，秘密 a, b それぞれのシェアから積 ab を秘密としたシェアをつくることを考える．加算と同じように，$q(x) = p_a(x) \cdot p_b(x)$ とすると，$q(0) = ab$ は成り立つが，多項式 q の次数が最大で $2t$ になってしまう．そこで，参加者間で，次数を減らす手続きをとる．そのために Lagrange 補間を使う．有限体 $\mathbb{F}$ 上の次数 k 以下の任意の多項式 $h(x)$ は，$\mathbb{F}$ 上の異なる $k+1$ 個の評価点 $(\alpha_i, h(\alpha_i))$ で一意に定まる．このとき，$k+1$ 個の元からなる任意の集合 $C \subseteq \mathbb{F}$ に対し，次数 k 以下の多項式 $\delta_i(x)$ が存在し，

$$h(x) = \sum_{\alpha_i \in C} h(\alpha_i) \delta_i(x)$$

と表すことができる．この多項式 $\delta_i(x)$ は，

$$\delta_i(x) = \prod_{\alpha_j \in C \setminus \{\alpha_i\}} \frac{x - \alpha_j}{\alpha_i - \alpha_j}$$

である．参加者の数が $n \geq 2t + 1$ を満たすとする．上記の事実より，先ほどの多項式 $q(x)$ に対し，$q(0)$ は n 個の適切な係数 $\{\beta_1, \ldots, \beta_n\} \subseteq \mathbb{F}$ を使って，

$$q(0) = \sum_{v=1}^{n} \beta_v q(\alpha_v)$$

と各 $q(\alpha_v)$ の線形結合で表すことができる．ここで，$\beta_v = \delta_v(0)$ である．各参加者 P_u は手元で計算できる $q(\alpha_u) = p_a(\alpha_u) \cdot p_b(\alpha_u)$ に対し，これを秘密とした t 次以下のランダム多項式 $p_u^*(x)$ を選び，そのシェア列 $[q(\alpha_u); p_u^*]_t$ の各シェア $p_v^*(\alpha_v)$ を P_v に配る．配られたシェアに対し，各 P_u は手元で $\sum_{v=1}^{n} \beta_v p_v^*(\alpha_u)$ を計算する．

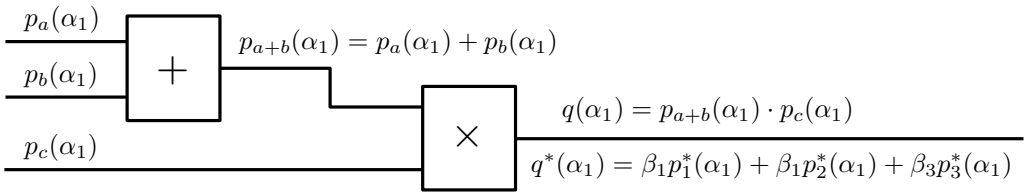

図 3　秘密分散による秘匿計算

これは秘密 $q(0) = ab$ に対し多項式 $q^*(x) = \sum_{v=1}^{n} \beta_v p_v^*(x)$ でつくられるシェア列 $[q(0); q^*(x)]_t$ の u 番目である．なぜならば，$q^*(x)$ は次数 t 以下であり，

$$q^*(0) = \sum_{v=1}^{n} \beta_v p_v^*(0) = \sum_{v=1}^{n} \beta_v q(\alpha_v) = q(0) = ab$$

である．このようにして，乗算素子においても，各参加者が秘密分散のシェアをつくり配ることで，$q^*(0) = ab$ を満たす次数 t 以下の多項式 q^* のシェアを計算できる．

以上より，すべてのワイヤに対し，ワイヤの値を秘密とした秘密分散のシェアを計算することができる．回路の出力ワイヤについても計算できており，各参加者は出力ワイヤに対応するシェアを他の参加者に伝えれば，すべての参加者が出力値を得ることができる．

図 3 は，$n = 3, t = 1$ の場合の，参加者 P_1 の計算の様子である．回路としては，入力 a, b, c に対し，$(a + b)c$ を出力するものである．乗算素子では，$q(\alpha_1)$ を求めた後，$p_1^*(0) = q(\alpha_1)$ を満たす多項式 p_1^* によるシェアをつくり，P_2, P_3 から $p_2^*(\alpha_1), p_3^*(\alpha_1)$ を受けとることで，$q^*(\alpha_1)$ を求めている．このシェアは，t 次以下の多項式 $q^*(x)$ のシェアであり，これより $q^*(0) = (a + b)c$ が復元される．

この方法は，乗算素子の計算において対話が必要であり，参加者数 n について $n \geq 2t + 1$ を満たす必要がある．秘密分散の性質より，t 人の参加者がシェアを持ち寄っても秘密が復元できない．制約は，$t < n/2$ と言い換えられる．つまり，秘匿計算において，（準正直な）敵対者の数が参加者の半数未満であれば，敵対者が結託しても，他の参加者の入力に関する情報を得ることはできない．逆に言うと，過半数の参加者が不正をしないことが必要であり，$n = 2$ である二者間の秘匿計算にはそのままでは適用できない．

3　悪意のある敵対者への対処

これまではプロトコルに従いながら相手の入力の情報を得ようとする準正直な敵対者に対して安全な秘匿計算プロトコルを考えてきた．ここでは，悪意のある敵対者に対処する方法をいくつか紹介する．

切り分け選択

目隠し回路による秘匿計算では，P_1 が目隠し回路をつくり，P_2 がそれを評価していた．悪意のある P_1 だと，目隠し回路自体を出鱈目につくる可能性がある．切り分け選択 (cut-and-choose) と呼ばれる方法では，P_1 は独立にたくさんの目隠し回路をつくり，P_2 がその一部を選び，正しくつくっているか確認し，その後，確認をしていない回路を使って計算を行う．正しさの確認は，P_1 が生成の際に使った乱数列を出すように言えばよい．P_2 はそれをもとに同じ回路が出力されるか検証できる．

素朴には，P_1 が独立に s 個の目隠し回路をつくり，P_2 がランダムに $s - 1$ 個を選んで確認し，残った 1 個を計算に使うという方法が考えられる．しかし，この方法では，P_1 が 1 個だけ不正につくったとき，確率 $1/s$ で不正に成功する可能性がある．この確率を，無視できるほどまでに小さくしたい．そのために，P_2 は $s - 1$ 個ではなく，各回路を確率 $1/2$ で選び，選ばなかった回路の出力に対し多数決をとったものを出力すればよい．こうすれば，P_2 の出力を変えるには，半数近い回路を不正につくっておく必要があり，その場合，不正が見つからない確率は $2^{-s/2}$ 程度であり，十分に小さい．

複数の回路を生成させるようにすると，P_1 は各回路に異なった入力を使う可能性が出てくる．それを防ぐために

は，入力の一貫性を保つ必要がある．その他にも，紛失通信プロトコルへの入力を不正なものにする可能性もあり，プロトコルをさらに強化する必要がある．

コミットメントとゼロ知識証明

次に，準正直な敵対者に対し安全なプロトコルを，悪意のある敵対者に対して安全にする一般的な方法を紹介する．そのアイディアは，準正直にしかプロトコルを実行できなくすることである．各参加者は，プロトコルの実行において，自身の入力に加え，乱数列を使う．そこで，コミットメントプロトコルにより，入力と乱数列を固定し，ゼロ知識証明により正しく実行していることを証明しながらプロトコルを進めるという方法である．

コミットメントプロトコルでは，送り手は入力を暗号化のように秘匿されたコミットメントとして，受け手に渡す．これがコミットフェーズである．開示フェーズでは，送り手は，コミットした内容を受け手に明らかにする．安全性要件は 2 つあり，コミットフェーズにおいて受け手は入力内容が分からないという秘匿性 (hiding) と，開示フェーズにおいて送り手は入力したもの以外に開けることができないという拘束性 (binding) である．

クラス NP に属する任意の言語に対し，ゼロ知識証明プロトコルを構成することができる．簡単に言えば，証拠が与えられると多項式時間で正しさを検証できるものに対して，正しさ以外の情報を漏らさずに正しいことを納得させることができる．準正直安全なプロトコルに対し，まず入力と乱数列をコミットメントとして相手に渡す．プロトコルの各ラウンドにおいて，相手に送るメッセージをつくった後，コミットメントを開示するための情報を証拠として，プロトコルの記述どおりに正しく計算していることのゼロ知識証明を行う．各ラウンドで送るメッセージは，それまでに受けとったメッセージと入力，乱数列から一意に定まるため，正しいメッセージ集合で構成される言語は NP に属する．プロトコルで対話をするたびにこのようなゼロ知識証明を行えば，各参加者は準正直にプロトコルを実行するしかない．また，各参加者が使う乱数列についても，公平なコイン投げになっていることを保証し，その情報は他の参加者に秘匿する必要がある．これは，コイン投げプロトコルで実現できる．

検証可能秘密分散

最後に，情報理論的な安全性を達成する方法を紹介する．準正直な安全性をもつ秘密分散にもとづいた秘匿計算の方法を紹介したが，それを発展させて悪意のある敵へ対処する方法がある．ここで登場するのが検証可能秘密分散 (verifiable secret sharing) である．通常の秘密分散では，秘密からシェアをつくり配る人も，集めたシェアから秘密を復元する人も，プロトコル通りに計算することを前提としていた．端的に言えば，検証可能秘密分散では，不正者がいても秘密の分散と復元に影響がないことを保証する．秘密をもつ人が不正にシェアを生成しても，復元するときに不正なシェアを出したとしても，復元結果が参加者間で異なることを防ぐ．不正に振る舞うと，本来の秘密とは異なる値に対してシェアをつくる可能性があり，また，そのような行為を防ぐことはできない．そこで，一度シェアを配れば，それが不正に生成されたとしても，そのシェアに対応する秘密の値がコミットされ，それ以外の値には復元できないことを保証する．秘密をもつ人が不正にシェアをつくる場合への対処が，特に工夫が必要な部分である．シェアを集めるときに不正に振る舞うことは，誤り訂正符号に対する誤りの挿入に対応する．Shamir の秘密分散は，Reed-Solomon 符号に対応しているため，シェアを集める際の不正は，Reed-Solomon 符号の復号アルゴリズムで対処できる．

ノート

目隠し回路は，Yao が 1982 年に示した方法である．秘密分散と紛失通信を組み合わせて実現した 1 節の方法は，Goldreich, Micali, Wigderson が 1987 年に示したものであり，GMW プロトコルと呼ばれる．秘密分散により情報理論的な安全性を実現した 2 節の方法は，Ben-Or, Goldwasser, Wigderson が 1988 年に示しており，BGW プロトコルと呼ばれる．

コミットメントとゼロ知識証明によって準正直安全なプロトコルを悪意のある敵対者に対し安全にする方法は，Goldreich, Micali, Wigderson が上記と同じ論文で提案しており，GMW 変換法と呼ばれている．