Vol.2013-DPS-154 No.5
Vol.2013-CSEC-60 No.5
2013/3/14

関数型暗号アプリケーションにおける適切な述語付与方式の検討

市川　幸宏[1,a)]　松田　規[1]　坂上　勉[1]

概要：岡本・高島の考案した関数型暗号は、論理演算子と属性で構成された述語を暗号化で利用する。この暗号アルゴリズムの特徴は、暗号化とアクセス制御の両方の機能を持つ。この暗号の応用として、我々はクラウドのような SaaS 型のファイル共有システムへの適用、さらにユーザ秘密鍵を IC カードへ格納するシステムの検討を行っており、処理性能の低いチップ上の適用を目指している。岡本・高島は CANS11 で上記暗号の高速化を実現しているが、さらなる高速化を希求している。
関数型暗号を適用したファイル共有システムは従来のものと異なり、アクセス制御を設定するユーザが、ファイル共有サーバ側の管理者から一般ユーザへと変更される。そのため、ファイル構成等を把握していない一般ユーザが、アクセス制御を実施するため、適切に暗号化を設定できない可能性がある。関数型暗号は、暗号化に用いる論理演算子の数と処理時間が比例するため、役職や所属などの階層構造を最大限活用し、より上位の属性を用いて述語を生成することで、効率的に暗号化することができる。そこで我々は、従来の関数型暗号アプリケーションよりも効率的に述語を付与する暗号化アプリを提案する。我々のアプリは、規模が数万ユーザを対象としたアプリの場合においても、数百 byte のバッファと深さ 1 で 2 桁のノードを持つ木の探索の処理時間を使って、関数型暗号の暗号化および復号時間を大幅に短縮することを実現する。

1. はじめに

Waters らによって提案された Functional Encryption(関数型暗号または高機能暗号) は，and や or の論理演算子で属性を連結した述語を暗号化で利用する。関数型暗号は、この述語による暗号化とアクセス制御の両方の機能を持つ暗号アルゴリズムである [1][2]。岡本・高島の提案した暗号も同様の機能を持ち、さらに not の論理演算子を述語で利用することが可能である [3][4][5][6][7]。これらの応用として、我々はクラウドのような SaaS 型のファイル共有システムへの適用、さらにユーザ秘密鍵を IC カードへ格納するシステムの検討を行っており、処理性能の低いチップ上の適用を目指している。岡本・高島は CANS11 で、暗号化もしくは復号の処理を O(1) で処理可能な方式を提案しており、高速化を実現した [7]。我々は、異なる方式で、さらなる高速化を希求している [8]。

関数型暗号を適用したファイル共有では、アクセス制御を設定するユーザが一般のファイル共有システムと異なるため、非効率な状態に陥る可能性がある。一般のファイル共有は、すべてを把握しているシステム管理者がアクセス制御を設定し、一般ユーザはそれを利用するだけであるが、関数型暗号では、一般ユーザの暗号化により、アクセス制御を設定するため、一般ユーザにとって便利になる一方、ユーザの中には、従来暗号と同様、ユーザ一人一人に対して、暗号化を設定する可能性がある。このような暗号化は必ずしも誤った利用とはいえないが、関数型暗号は、従来の公開鍵暗号のように復号を許可されたユーザ数と処理時間が比例するのではなく、論理演算子の数と処理時間が比例するため、役職や所属といった属性の階層構造を最大限活用し、より上位の属性を用いれば、効率的に暗号化することが可能となる。そこで我々は、従来の関数型暗号アプリケーションよりも適切な述語を付与する効率的な暗号化アプリケーションを提案する。我々のアプリケーションは規模が数万ユーザを対象とした場合においても、数百 byte のバッファと深さ 1 で 2 桁のノードを持つ木の探索の処理時間を使って、関数型暗号の暗号化および復号時間を大幅に短縮することを実現する。また、提案方式を導入した関数型暗号ファイル共有システムを考慮した場合、構成等を大きく変化させないため、ほとんどの関数型暗号アプリケーションと親和性が高いと考えている。

[1] 三菱電機株式会社
〒 247-8501 神奈川県鎌倉市大船 5-1-1.Mitsubishi Electric Corporation,Information Technology R & D Center,5-1-1 Ofuna Kamakura, Kanagawa, 247-8501, Japan.
[a)] Ichikawa.Sachihiro@dp.MitsubishiElectric.co.jp

本稿では、上記アプリケーションが有効となる状況を説明し、アプリケーションのインターフェイスおよびシステム構成について説明する。考察では、本アプリの性能見積もりおよび、関数型暗号を適用したファイル共有システム特有の問題である新規ユーザ追加によるアクセス制御の範囲の変化について述べる。

2. 適切な述語付与について

2.1 関数型暗号の概要

関数型暗号は、Setup、KeyGen、Enc、Dec の 4 つのアルゴリズムで構成されている。本稿で関数型暗号はユーザ秘密鍵に属性を、暗号文に述語を入れる Ciphertext-Policy 型である。

Setup 鍵生成サーバは公開パラメータ pk とマスタ秘密鍵 sk を生成する。pk は暗号化やユーザ秘密鍵生成の計算に必要で、すべてのユーザに公開される。sk はユーザ秘密鍵の生成に必要で公開しない。

KeyGen 鍵生成サーバは Setup で生成した公開パラメータ pk およびマスタ秘密鍵 sk、ユーザ秘密鍵に埋め込む属性、3 つの情報を入力し、ユーザ秘密鍵 k を生成する。属性とは、「人事部, 部長, 田中」のような組織情報や職制情報、氏名などで表現した文字列である。ユーザ秘密鍵 k は暗号文の復号に必要で、対象となるユーザ以外には公開しない。

Enc ユーザは公開パラメータ pk と文章 m、文章を暗号化する述語の 3 つを入力し、暗号文 c を生成する。述語の例としては、「人事部 and 部長」のような属性と論理演算子を連結した文字列である。

Dec ユーザは公開パラメータ pk と暗号文 c、ユーザ秘密鍵 k の 3 つを入力し、暗号文 c を復号する。暗号文の述語とユーザ秘密鍵の属性が一致している場合、復号に成功し、そうでない場合、復号に失敗する。

上記の暗号アルゴリズムを用いて、ファイル共有システムに組み込む。次にアルゴリズムの処理フローについての概要を述べる。

2.2 各アルゴリズムの詳細

ここでは、スパンプログラムの関数型暗号のアルゴリズムについて、述語の論理式の数と速度が比例することを示す [6]。

Setup 公開パラメータ pk とマスタ秘密鍵 sk を生成するアルゴリズムである。Setup アルゴリズムは、双対ペアリングベクトル空間のパラメータ param と、$t = 0, \ldots, d$(d は 1 以上の整数) の各 t についての正規直交基底である基底 $\mathbb{B}_t$ 及び基底 $\mathbb{B}_t^*$ を生成する。そして、パラメータ param、基底 $\mathbb{B}_t$ を公開パラメータ pk とし、基底 $\mathbb{B}_t^*$ をマスタ秘密鍵 sk とする。なお、基底 $\mathbb{B}_0$ は、3 つの基底ベクトル $b_{0,1}, b_{0,3}, b_{0,5}$ を持ち、基底 $\mathbb{B}_0^*$ は、3 つの基底ベクトル $b_{0,1}^*, b_{0,3}^*, b_{0,4}^*$ を持つ。また、$t = 1, \ldots, d$ の各 t についての基底 $\mathbb{B}_t$ は、基底ベクトル $b_{t,1}, \ldots, b_{t,n_t}, b_{t,3n_t}$ を持ち、基底 $\mathbb{B}_t^*$ は、基底ベクトル $b_{t,1}^*, \ldots, b_{t,n_t}^*, b_{t,2n_t+1}^*, \ldots, b_{t,3n_t}^*$ を持つ。つまり、基底 $\mathbb{B}_t$ は $n_t + 1$ 個、$\mathbb{B}_t^*$ は $2n_t$ 個 (n は 1 以上の整数) の基底ベクトルを持つ。

KeyGen ユーザ秘密鍵 k^* を生成するアルゴリズムである。KeyGen アルゴリズムは、マスタ秘密鍵 sk に含まれる基底 $\mathbb{B}_t^*$ を用いて、要素 k_0^* と、$t = 0, \ldots, d$ の各 t についての要素 k_t^* を持つユーザ秘密鍵 k^* を生成する。

$$k_0^* := (\delta, 0, 1, \varphi_0, 0)\mathbb{B}_0^*$$
$$k_t^* := (\delta \overrightarrow{v_t}, 0^{n_t}, \overrightarrow{\varphi_t}, 0)\mathbb{B}_t^* \quad (1)$$

$\delta, \varphi_0, \overrightarrow{\varphi_t} := \varphi_{t,1} \ldots, \varphi_{t,n_t}$ は、乱数値である。$\overrightarrow{v_t} := v_{t,1}, \ldots, v_{t,n_t}$ は、ユーザ秘密鍵 k^* が与えられるユーザの属性関連情報である。式 2 で、秘密鍵に設定する属性をベクトルで示す。組織情報を変数 A、職制情報を変数 B、「PD(人事部), M(部長)」を表現する。

$$(A, (1, PD))$$
$$(B, (1, M)) \quad (2)$$

式 3 はユーザ秘密鍵を示す。

$$k_0^* := (\delta, 0, 1, \varphi_0, 0)\mathbb{B}_0^*$$
$$k_A^* := (\delta(1, PD), 0^{n_A}, \overrightarrow{\varphi_A}, 0)\mathbb{B}_A^*$$
$$k_B^* := (\delta(1, M), 0^{n_B}, \overrightarrow{\varphi_B}, 0)\mathbb{B}_B^* \quad (3)$$

Enc 暗号化データ c を生成するアルゴリズムである。公開パラメータ pk に含まれる基底 $\mathbb{B}_t$ を用いて、暗号化データ c の要素 c_0 と，$t = 1, \ldots, L$（L は、d 以下の整数）の各 t についての要素 c_t を生成する。

$$c_0 := (-s_0, 0, \zeta, 0, \eta_0)\mathbb{B}_0$$
$$c_t := (s_t \overrightarrow{e}_{t,1} + \theta_t \overrightarrow{x}_t, 0^{n_t}, 0^{n_t}, \eta_i)\mathbb{B}_t \quad (4)$$

$\overrightarrow{e}_{t,1}$ は、n_t 個の要素を持ち、先頭要素が 1 で残りの要素が 0 であるベクトルを示す。$s_0 = \Sigma_{i=1}^{L} s_i$ である。$\zeta, \theta_t, \eta_0, \eta_t$ は、乱数値である。$x_{t,1}, \ldots, x_{t,n_t}$ は、暗号化データ c を復号可能なユーザの属性関連情報である。$(z_1, \ldots, z_N)\mathbb{B}_t := \Sigma_{i=1}^{N} z_i b_{t,i}$ である。

Enc アルゴリズムは、公開パラメータ pk に含まれるパラメータ param を用いて、暗号化データ c の要素 c_{d+1} を生成する。

$$c_{d+1} := e(g, g)\zeta \cdot m \quad (5)$$

g は、パラメータ param に含まれる情報であり、双対ペアリングベクトル空間を構成する群 G の要素で

ある。m は、メッセージである。e(g, g) は、要素 g と要素 g とについてのペアリング演算である。「PD(人事部) and M(部長)」の暗号文 c をそれぞれのベクトル $\vec{x}_A = (A, (PD, -1)), \vec{x}_B = (B, (M, -1))$ を用いて示す。

$$\begin{aligned} c_0 &:= (-s_0, 0, \zeta, 0, \eta_0)\mathbb{B}_0 \\ c_A &:= (s_A \vec{e}_{A,1} + \theta_A(A, (PD, -1)), 0^{n_t}, 0^{n_t}, \eta_i)\mathbb{B}_t \\ c_B &:= (s_B \vec{e}_{B,1} + \theta_B(B, (D, -1)), 0^{n_t}, 0^{n_t}, \eta_i)\mathbb{B}_t \end{aligned} \tag{6}$$

式 7 は and を示す行列式 M である。

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7}$$

Dec Dec アルゴリズムは、暗号化データ c がユーザ秘密鍵 k^* を用いて m′ を抽出する。

$$m' := c_{d+1}/(e(c_0, k_0^*) \cdot \Pi_{t=1}^{L} e(c_t, k_t^*)) \tag{8}$$

$e(c_0, k_0^*)$ は、要素 c_0 とユーザ秘密鍵 k_0^* のペアリング演算を示し、$e(c_t, k_t^*)$ は、要素 c_t とユーザ秘密鍵 k_t^* も同様、ペアリング演算を示す。$t = 1, \ldots, L$ の各 t について、ユーザ秘密鍵 k^* における要素 k_t^* に設定された属性等 $(\vec{v}_t)$ と、暗号化データ c における要素 c_t に設定された属性等 $(\vec{x}_t)$ が対応する場合、$m' = m$ になる。属性等 $(\vec{v}_t)$ と属性等 $(\vec{x}_t)$ が対応する場合、$\vec{v}_t \cdot \vec{x}_t = \Sigma_{i=1}^{n_t v_{t,i} \cdot x_{t,i} = 0}$ になる。

式 8 を解くために 3 つに分けて計算を実施する。

(1)$e(c_0, k_0^*)$ に関して、ペアリング演算 $e(sg, tg) = e(g, g)^{st}$ であるため、$e(c_0, k_0^*) = e(g, g)^{Y1}$ になり、$Y1 = -s_0\delta + \zeta$ である。

(2)$\Pi_{t=1}^{L} e(c_t, k_t^* = (g, g)^{Y2})$ になり、$Y2 = \Sigma_{i=1}^{L}(s_i + \vec{v}_t \cdot \vec{x}_t) = \Sigma_{i=1}^{L}(s_i) + \Sigma_{i=1}^{L} \vec{v}_i \cdot \vec{x}_i$ である。したがって、$\Sigma_{i=1}^{L} \vec{v}_t \cdot \vec{x}_t = 0$ の場合、$Y2 = \Sigma_{i=1}^{L}(s_i)$ になる。

(3)(1) と (2) を合わせた $e(c_0, k_0^*) \cdot \Pi_{t=1}^{L} e(c_t, k_t^* = e(g, g)^{Y3})$ は、$\Sigma_{i=1}^{L} \vec{v}_t \cdot \vec{x}_t = 0$ の場合、$Y3 = -s_0\delta + \zeta + \Sigma_{i=1}^{L}(s_i)$ になる。Enc で述べたように $s_0 = \Sigma_{i=1}^{L} s_i$ であるため、$Y3 = \zeta$ だけが残る。$e(c_0, k_0^*) \cdot \Pi_{t=1}^{L} e(c_t, k_t^*) = e(g, g)^{\zeta}$ は、$c_{d+1} := e(g, g)^{\zeta} \cdot m$ であるから、$\Sigma_{i=1}^{L} \vec{v}_t \cdot \vec{x}_t = 0$ の場合、$m' = m$ になる。

上記のように属性と論理演算子の組み合わせにより、式 6 の c の要素が増加する。この式は暗号文を示し、論理演算の組み合わせ、つまりに述語の長さに比例して要素が増加するため、Enc と Dec において、処理時間が増加する。述語の長さと速度が比例することを示せた。次に一般の論理演算を省略する方式について述べる。

2.3 論理演算の省略について

一般に論理演算の省略方法として用いられるのが、シャノンの展開定理やカルノー図、クワイン・マクラスキー法、ブール代数の公理における吸収律を用いる方法などである。しかし、これらの方式を用いることで、省略することが可能な論理式は、一般のアクセス制御で用いられる条件とは大きく異なる。具体的には、シャノンの展開定理を用いて省略可能な以下の式を考える。

$$A \text{ and } \overline{B} \text{ and } \overline{C} \text{ or } \overline{A} \text{ and } B \text{ and } \overline{C} \text{ or } \overline{A} \text{ and } \overline{B} \text{ and } C \text{ or } A \text{ and } B \text{ and } C$$

上記式は、 以下の式に省略可能であるが、

$$A \text{ and } B \text{ or } \overline{B} \text{ and } \overline{A}$$

ファイル共有システムにおけるアクセス制御に対して、上記の類似例を以下の式に示す。

人事部 and not 課長 and not 田中 or not 人事部 and 課長 and not 田中 or not 人事部 and not 課長 and 田中 or 人事部 and 課長 and 田中

このようなアクセス制御は、実運用上で設定される可能性が低いと考えられる。一般のアクセス制御では、異なる属性に対して、アクセス権の追加を目的として、or をつなげて制御するものが多く、また、このような同じ属性を複数回用いる制御の可能性は少ない。そこで我々は関数型暗号に用いられる属性の意味の関係性、特に階層構造に着目して、より上位の属性を用いて or 論理演算を省略する方式を提案する。階層構造は木構造で表現できる。

3. 提案方式

提案方式は、Enc アルゴリズムで入力される述語について省略し、Enc と Dec の高速化を試みる。

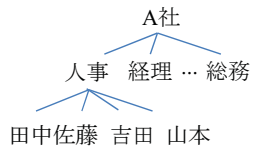

図 1 rooted tree 例

関数型暗号で用いられる属性の階層構造は、一般の rooted tree(以下、木) で表現できる。図 1 の例は、A 社をルートとし、その下の内部ノードには部名、リーフには、その部に組する社員の氏名が示されている。このような組織情報を用いて、ユーザが暗号化に利用する述語を省略する。述語が以下の場合、

田中 or 佐藤 or 吉田 or 山本 and 課長

「田中 or 佐藤 or 吉田 or 山本」は、さらに上位の階層情報「人事」でまとめることが可能である。よって以下に

なる。

人事 and 課長

この変換によって、論理演算子 or が 3 つ、and が 1 つの式を、and が 1 つの式に変換することができ、暗号化と復号の速度を約 1/4 にすることが可能である。さらに、岡本・高島の考案した暗号は、not の論理演算子を述語で用いることが可能である [6][7]。Waters らの暗号では、上記条件のような上位属性でまとめられる場合の省略しかできないが、not を活用することで、さらに異なる条件であっても省略することができる。述語が以下の場合、

田中 or 佐藤 or 吉田 and 課長

「田中 or 佐藤 or 吉田」は、さらに上位の階層情報「人事」でまとめることが可能である。よって以下のように変換できる。

人事 and not 山本 and 課長

この省略変換により、論理演算 or が 3 つと and が 1 つの式から、and が 2 つの式に変換することができた。not に関しては、not が付与されない暗号文である式 9、not が付与された暗号文である式 10 で示すとおり、暗号文 c としての情報は異なるが、暗号文 c の要素自体は増加せず、演算速度は変わらない。

$$c_i := (s_i \overrightarrow{e}_{i,1} + \theta_i \overrightarrow{v}_{i,1}), 0^{n_t}, 0^{n_t}, \eta_i)\mathbb{B}_t \quad (9)$$

$$c_i := (s_i \overrightarrow{v}_{i,1}, 0^{n_t}, 0^{n_t}, \eta_i)\mathbb{B}_t \quad (10)$$

また、暗号化に関して or と and は、同等の計算量であるため、暗号化と復号の速度を 1/2 にできる。このような工夫をすることで、関数型暗号はより高速に処理することが可能となり、Android や iPhone などの小型端末や演算処理の非力な IC カードへの実装を期待させる。

4. 提案アプリケーション

一般に関数型暗号を SaaS 型のファイル共有システムへ適用した場合のシステム構成図を図 2 に示す。

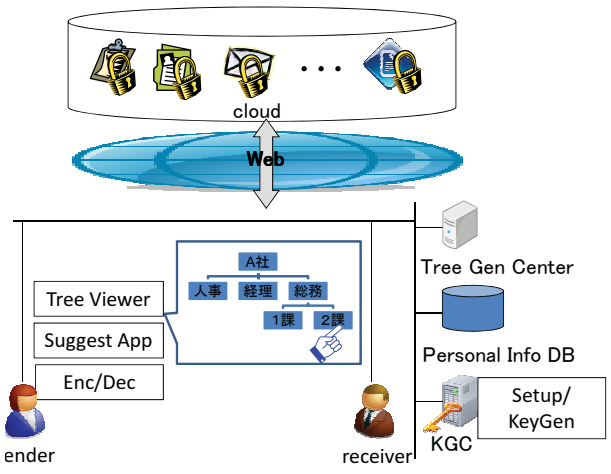

図 2　関数型暗号を適用したファイル共有システム

sender　クラウドへ暗号化データを送付するユーザである。Tree Viewer は、組織情報から暗号化対象になる属性の見える化と属性を選択させるアプリケーションであり、その選択された情報と論理演算子から述語を生成する。Suggest App(提案アプリケーション) は生成した述語を最適な形に変換し、Enc/Dec へ渡す。Enc/Dec は、データを暗号化し、クラウドへ送付する。Enc/Dec は、関数型暗号を実現する上では必要不可欠であり、別の端末（プロキシ PC など）にインストールして利用する運用も考えられる。Tree Viewer は必ずしも必要ではないが、これを利用しない前提の場合、ユーザがフリーフォーマットで述語を生成しなければならず、運用が困難になると考え、ユーザビリティを考慮するのであれば、必要不可欠なアプリケーションである。Suggest App は Tree Viewer に組み込まれることを想定している。

receiver　クラウドから暗号化データを取得する、もしくは、鍵サーバから暗号文を復号するために必要なユーザ秘密鍵を取得するユーザで、その秘密鍵を用いて暗号文を復号することができるユーザである。図には明記していないが、Enc/Dec を持ち、関数型暗号を実現する上では必要不可欠である。復号は別の端末（プロキシ PC など）を利用してもよい。

cloud　暗号文を保管するサーバである。暗号化保管に必要不可欠である。

Personal Info DB　組織内部で格納している人事情報を管理するサーバであり、LDAP や AD で運用されることが多い。上記システムにおいて、補助の役割を担っている。sender および receiver、KGC、Tree Gen Center が、属性のすべてを把握している場合、本サーバは不要であるが、多くの組織ではすでに上記 DB を持っていることが多く、そのままの適用が望ましい。

Tree Gen Center　Personal Info DB から人事情報を取得し、述語を作成するために必要な情報を木構造としてユーザに提供するサーバである。上記システムにおいて、補助の役割を担っている。sender が容易に述語を生成できる場合は不要であるが、ユーザに負荷をかけない運用にするため、利用が望ましい。

KGC　マスター鍵ペア生成機能 (Setup) やユーザ秘密鍵生成機能 (KeyGen) を持つサーバである。関数型暗号を実現する上では必要不可欠である。

上記システム構成が関数型暗号を適用した場合、一般と考えられ、各ユーザに提案方式を実現するアプリケーションがインストールされる。ユーザビリティ向上のため、図 2 に示す木構造から属性を選択し、論理演算子を属性と属性の間に付与し、述語を生成する。

次に Tree Gen Center から生成された木が、深さ 3 以上の木である場合の述語省略操作について述べる。例では深

さ 3 であるが、提案方式は深さについて制限を設けない。この操作を繰り返すことで、最小単位の述語を構成することが可能である。

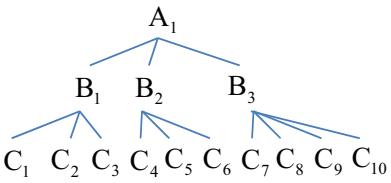

図 3 depth=3 の木の例

(1) ユーザが選択した属性を子供とし、その親の情報を保管する。
(2) その親が複数保管された場合、親の一致・不一致を確認する。
(3) その親の持つ子供の数を保管する。
(4) 一定のしきい値以上であれば、同じ親を持つ複数の子供は、親一つにまとめる変換を行う。
(5) 変換した場合かつ、親の持つ子供の数と置き換えられる子供の数が一致しない場合、その親の持つ子供の中から、ユーザの選択していない子供を抽出し、その子供に and not を付与し、述語にする。
(6) 上記を繰り返し、一定のしきい値をどの要素も超えなければ操作終了とする。

しきい値に関して、下記の式を用いて判断する。

$$Z \geq n/2 + 1$$

Z は操作 2 で一致した数、n は親が持つ子供の数となる。図 3 を用いて、以下の述語である場合、処理の流れを示す。

$$C_1\ or\ C_2\ or\ C_3\ or\ C_4\ or\ C_5\ orC_6\ or\ C_7\ or\ C_8\ or\ C_9$$

1) 属性 C_1 の親は B_1 である。2) 親を格納するバッファには、複数保管されていないため、この操作は行わない。3) バッファに格納した親の子供の数を取得する。B_1 の子供の数は $n_{B1} = 3$ となる。3) までの操作を属性が読めなくなるまで実施する。

1) 属性 C_2 の親は B_1 である。2) 親を格納するバッファには、B_1 が 2 つ格納されているため、一致か不一致か確認する。一致しているため、一致した情報 $Z_{B1} = 2$ をバッファに格納する。3)B_1 の子供の数はすでに取得済みであるため次の処理を実施する。これを順に実施していくと、親情報バッファには B_1, B_2, B_3、子供の数は $n_{B1} = 3, n_{B2} = 3, n_{B3} = 4$、一致した情報は $Z_{B1} = 3, Z_{B2} = 3, Z_{B3} = 3$ となる。4) 取得した値を用いて、しきい値計算する。

(B1)$3 \geq 3/2 + 1$
(B2)$3 \geq 3/2 + 1$
(B3)$3 \geq 4/2 + 1$

4) すべて条件を満たしているので、

$$B_1\ or\ B_2\ or\ B_3$$

に変換する。5) 条件を満たしているので、

$$B_1\ or\ B_2\ or\ B_3\ and\ not\ C_{10}$$

になる。6) バッファをクリアし、再度、1) から操作する。1) 属性 B_1 の親は A_1 である。2) 現状、親を格納するバッファには、複数保管されていないため、この操作は行わない。3) バッファに格納した親の子供の数を取得する。A_1 の子供の数は $n_{A1} = 3$ となる。3) までの操作を属性が読めなくなるまで実施する。

1) 属性 B_2 の親は A_1 である。2) 親を格納するバッファには、A_1 が 2 つ格納されているため、一致か不一致か確認する。一致しているため、$Z_{A1} = 2$ をバッファに格納する。3) すでに操作しているため次の処理を実施する。これを順に実施していくと、親情報バッファには A_1, B_3、子供の数は $n_{A1} = 3, n_{B3} = 4$、一致した情報は $Z_{A1} = 3, Z_{B3} = 1$ になる。

4) ここで、上記で取得した値を用いて、しきい値計算する。

(A1)$3 \geq 3/2 + 1$
(B3)$1 \geq 4/2 + 1$

[A1] のみ条件を満たしているので、

$$A_1\ and\ not\ C_{10}$$

に変換する。5) 条件は満たしていない。6) の条件は満たすため、バッファをクリアし、再度、1) から操作する。次の操作では、どの要素もしきい値を満たさないため終了する。

5. 評価および考察

5.1 方式の有効性

提案方式は、or の論理演算子を用いた述語に関して、上位属性および not 演算子を用いて述語を短くできる。and の論理演算子の省略は、従来方式であるシャノンの展開定理などを用いる。or の論理演算子の省略について、現状のアクセス制御の多くは、権限を追加する運用に従い、対象ユーザを or でつなげる。and を用いる場合は、すでについている条件に対して、追加で異なる属性を付与し、厳しいアクセス制御を付与する場合に用いられることが多く、例えば、経理部 and 部長などである。現状、本システムと一般に運用されているアクセス制御とは、同様の運用が考えられるため、or を複数回用いて権限を追加し、and を単一もしくは、数回利用することが多いと想定される。よって、提案方式が or に対して省略できる機能は有用であると考える。

提案方式にかかる時間およびメモリ量を概算する。35,000 名ほどが組する企業があると想定し、木構造の深さを 4 と

し、それぞれのノードの数を、深さ 1 のノードの数を 40、深さ 2 のノードの数を 30、深さ 3 のノードを 3、葉を 10 として考える。バッファとしてキャッシュしなければならない親の情報は $40 * 4$byte、子供の数は $40 * 1$byte である。ある深さのノード数を N とした場合、一致不一致および述語変換、not を付与する場合の子供の検索に関しては O(N) で処理する。一つのノード数が例のように 2 桁程度であれば、提案方式の方に魅力があると考える。最悪値は、木構造に依存するため、ある深さのノード数が膨大なデータを扱う場合である。回避方法として、別のデータ構造を用いる方式の検討も必要である。

まとめると、上記規模の 200byte 程度のバッファと木の最大ノード数に比例する O(N) となる。論理演算が増加するたびに、ペアリングの計算をするよりも処理が高速になるのは自明である。また、ペアリングの計算は、現状のコプロで高速処理できないため、このような回避策も有用ではないかと考える。

5.2 安全性の変更

提案方式では、従来の論理式省略とは異なり、意味を考慮したため、厳密には、アクセス制御条件を変更してしまう問題がある。例えば、D を親として、その子供を A、B、C にする $A, B, C \in D$ の条件で、述語を A or B or C にする。本方式を用いて D に変換し、D の暗号文を保管する。その後、新規ユーザの追加により、条件が上記から $A, B, C, a \in D$ へ変化する。その場合、変換前の暗号文はユーザ a の秘密鍵を用いても復号できないが、変換後であれば、ユーザ a の復号を許可することになる。ユーザは、これを許可するか否か考慮しなければならない。特に通信路の暗号化であれば、提案方式は問題ないが、今回のファイル共有システムでは、問題になる可能性がある。あらかじめ述語を D とした場合、現在の条件と未来の条件が異なるため、提案方式とは逆の方式を用いて、D から A or B or C へ変換するという考えもある。これらを考慮し、アプリとして、ユーザに確認を促す必要があると考える。

6. おわりに

関数型暗号、特に岡本・高島暗号に対して、述語を適切に付与するためのアプリケーションを提案した。岡本・高島暗号の特徴である not と、述語の階層構造を利用して論理演算子の or を省略し、述語を短くすることで、暗号化と復号の処理の高速化を実現した。提案アプリケーションは、一般的に構成される関数型暗号システムに少しだけのバッファと検索時間を費やすことで、大きな高速化を図ることが可能である。

参考文献

[1] Brent Waters:*Functional encryption:beyond public key cryptography*, Power Point Presentation (2008). http://userweb.cs.utexas.edu/ bwaters/presentations/files/functional.ppt

[2] Dan Boneh, Amit Sahai, Brent Waters:*Functional Encryption: Definitions and Challenges*, Theory of Cryptography. LNCS (2011),Volume 6597/2011.

[3] T.Okamoto, K.Takashima:*A geometric approach on pairings and hierarchical predicate encryption*,Poster session,EUROCRYPT (2009).

[4] T.Okamoto, K.Takashima:*Hierarchical predicate encryption for inner-products*, ASIACRYPT (2009). LNCS, Volume 5912/2009.

[5] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, Brent Waters:*Fully Secure Functional Encryption:Attribute-Based Encryption and(Hierarchical)Inner Product Encryption*, EUROCRYPT (2010).LNCS, Volume 6110/2010.

[6] T.Okamoto, K.Takashima:*Fully Secure Functional Encryption With General Relations from the Decisional Linear Assumption*, CRYPTO (2010). LNCS, Volume 6223/2010.

[7] T.Okamoto, K. Takashima:*Achieving short ciphertexts or short secret-keys for adaptively secure general inner-product encryption*, CANS(2011). pp.138-159.

[8] N.Matsuda:*Efficient Smart-Card-based Decryption in Functional Encryption*, SCIS(2013).