Computer Security Symposium 2017
23 - 25 October 2017

関数型暗号を適用したIoT機器データ活用方式の一考察

田村 桜子[†1]　永井 彰[†1]

概要：今後IoT機器が多様化していくと，IoT機器が取得したセンサデータを，IoT機器同士が活用することも考えられる．その際に，IoT機器が取得したセンサデータが個人情報などの場合，悪意ある第三者への漏洩の防止やセンサデータの改ざん防止のためには，センサデータの暗号化が必要である．しかし，従来方式では，鍵管理が煩雑となることなどが課題となると考えられる．本稿では，暗号化時に属性情報を組み込むことが可能な暗号化方式である関数型暗号をIoT機器のセンサデータ活用を検討し，提案方式の有効性について最も懸念されるIoT機器における関数型暗号の性能面について評価を行い，提案方式の有効性を示した．

キーワード：IoT，Internet of Things，関数型暗号，センサデータ，制御

A Study of applying Functional Encryption to the Internet of Things

Sakurako Tamura[†1]　Akira Nagai[†1]

Abstract: In few year, more varions IoT devices appear, and IoT devices also can utilize date sensing from another IoT devices. To prevent leakage of sensor data such as personal information, encryption of the sensor data is necessary. However, conventional methods, such as common key or public key, apply to IoT systems that consist of many IoT devices, a management of keys becomes complicated. In order to solve this problem, we suggested that functional encryption algorithm, that enables data encryption and decryption with flexible conditions, apply to IoT systems. We showed the effectiveness of the proposed method in terms of the performance aspect of functional encryption which is the most concerned aspects.

Keywords: IoT，Internet of Things，Functional Encryption，sensor data, control

1. はじめに

2020年までにインターネットに接続されるモノは，およそ204億台に達すると予測されており[1]，今後多様なIoT (Internet of Things) 機器が市場に出回り，各機器が取得するセンサデータの種類も拡大すると予測できる．センサ機器からスマートフォンまで，様々なモノがIoT機器と呼ばれるが，本稿においては，IoT機器とは主にセンサ機器のようなゲートウェイに接続する機器に限定する．そのような環境の中で，IoT機器同士がセンサデータの授受を行い，受け取ったセンサデータを活用するような使い方も可能になる．現在のIoT機器のセンサデータの活用方法としては，工場などで各IoT機器から収集したデータをクラウド上で収集して，収集した膨大なデータを基にシステム全体の効率化に活用，各機器の状態を管理するという使い方が主である．今後多様な機器がインターネットにつながることを想定すると，センサデータから解析を行うだけでなく，IoT機器同士がセンサデータを授受し，IoT機器が他のIoT機器と連携して動作したり，特定のセンサデータをIoT機器の制御に活用するということも実現できる．例えば，血圧計や体重計などのIoT機器が取得したデータを，家電製品など他の機器が活用することも考えられる．メーカの異なるIoT機器間での連携をIoT機器にチップを組み込み，IoT機器間でのアクセス制御をクラウド側で実現するような検討例もある[2].

前述のようにセンサデータをIoT機器の制御などに利用する場合，制御するIoT機器が受信するセンサデータを改ざんされると，IoT機器がIoT機器の所有者が想定しない挙動をする可能性がある．また，センサデータ自体が個人情報などに紐付く情報であれば，センサデータ自体の情報漏洩を引き起こす可能性もある[3]．そのため，センサデータを暗号化して保管し，特定の機器へのみ閲覧可能とする技術が必要となる．

特定のIoT機器でのセンサデータの閲覧を実現するためには，暗号化や復号に利用する鍵をIoT機器ごとに共有し，共有した鍵により暗号化したデータを送信，共有した鍵により受信したデータを復号することが必要となる．例えば，共通鍵方式では，暗号化側，復号側で同様の鍵を共有する必要があり，公開鍵方式では，暗号化側が，復号側の秘密鍵と対になる公開鍵を共有する必要がある．IoT機器を利用するような場合，1つのIoT機器システムには管理されているIoT機器が数千単位で存在していることが珍しくない．よって，機器の半数がやり取りしようとしても，数百単位の鍵を各IoT機器で管理する必要が生じる．しかし，IoT機器は記憶容量が小さく，また記憶容量を増やすことも容易ではない．よって，共有先の復号鍵をすべて各機器

†1 NTTセキュアプラットフォーム研究所
NTT Secure Platform Laboratories

で保有することは現実的ではないといえる．また，各 IoT 機器の鍵を管理するサーバを配置し，鍵利用時に逐次送付する方式も考えられるが，数千単位の機器の鍵配布状況を把握し適切な鍵を配布するというのはシステム自体が煩雑になると考えられる．

今回我々はこれらを解決する方式を検討する上で，復号鍵に属性情報を含んだ条件式を組み込むことが特徴である，関数型暗号に着目した．関数型暗号の特徴を利用することにより，複数の IoT 機器と暗号化したセンサデータを共有する際に，IoT 機器は，複数のセンサデータが閲覧可能という情報を含んだ条件式により生成した復号鍵を 1 つ保有すればよい．よって，この特徴を利用することにより，膨大な IoT 機器における鍵の管理が容易になると予想したためである．

本稿では，関数型暗号を適用した，IoT 機器のセンサデータの利活用を行う方式の提案と，本提案方式の有効性を検討した. 2 章においては, 本稿で扱う用語や技術の定義，3 章では提案方式の詳細について, 4 章では提案方式を実現する上で最も懸念される IoT 機器における関数型暗号の性能について測定を行った．

2. 準備

本章では，本稿で用いる用語の定義，適用技術の詳細説明を示す．

2.1 IoT 機器システム

IoT 機器システムとは, 図 1 に示すように, IoT 機器, IoT GW，IoT クラウドの三者で構成をされ，各機器間で，センサデータのやり取りを行うシステムとする．

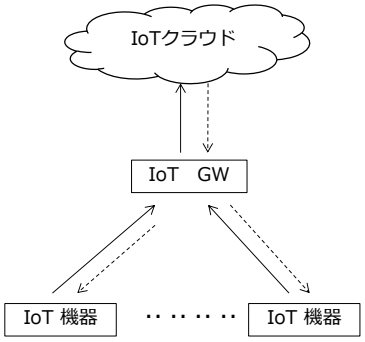

図 1　本稿で想定する IoT 機器システムモデル

IoT 機器システムを構成する機器の機能について，以下に詳細に示す．

❑　IoT 機器

本稿において IoT 機器とは主にセンサ機器のようなゲートウェイに接続する機器に限定する．IoT 機器は bluetooth や LPWA (Low Power Wide Area) などの消費電力の少ない無線通信を利用し，IoTGW へ接続する．

❑　IoT GW

IoT GW (gateway)とは，IoT 機器をネットワークに接続する中継機器である．IoT 機器とは，低消費無線通信である bluetooth や LPWA により通信を行い，センサデータの授受を行う．IoT クラウドとは，IP 通信を行い，センサデータの授受を行う．

❑　IoT クラウド

IoT GW を経由して送信されたセンサデータを保管する．

2.2 関数型暗号

関数型暗号とは，公開鍵暗号をより高度にした暗号方式であり，暗号文，属性鍵に属性情報や条件式を導入することで，暗号/復号のロジックを規定している．この暗号と復号のメカニズムには，AND ゲート，OR ゲート，NOT ゲート，しきい値ゲートにより表現されるすべての関係式を組み込むことが可能である[5]．関数型暗号には，暗号文に属性情報，復号鍵に条件式を組み込む，Key Policy 方式と，暗号文に条件式，復号鍵に属性情報を組み込む，Cipher text Policy 方式がある．本稿では，復号鍵に属性情報の OR 条件式を持たせることで，復号する機器側で保有する鍵の個数を削減することができると考え，Key Policy 方式を用いる．暗号アルゴリズムとしては，CCA 安全性を実現するために，Boneh-Katz 変換[6]を用いた OT-10[4] になっている．ここでは，Key Policy 方式の関数型暗号に用いるパラメータ, 各処理の概要を述べる. 特に 4 章にて性能評価を行う，関数方暗号の暗号化処理に関してはアルゴリズムを記述する．処理の詳細は[4]の文献を参照されたい．

○　必要なパラメータ，定義

mk ：マスター鍵
pk ：公開パラメータ
sk ：復号鍵
$\mathbb{S}$ ：鍵識別子(述語$\mathbb{S} \coloneqq (M,\ \rho)$)
M ：$l \times m$行列
Γ ：属性情報
m ：平文
$c_{t\Gamma}$ ：暗号文
1^{λ} ：セキュリティパラメータ
d ：属性空間の次元の配列数
$\vec{n}$ ：d個の異数空間の次元
$\mathbb{B}$ ：行列の配列
$\mathbb{B}_t$ ：楕円点からなる$(3t+1) \times (3t+1)$行列
Pub ：Boneh-Katz 変換におけるカプセル化パラメータ
g_T ：双対基底
$S_{encap}(K)$: Mac 鍵$mackey$を生成し, コミットメントcom, デコミットメント$decom$を生成

$KDF(K)$ ：シード K より，共通鍵を生成

$SE_K(M)$ ：K をシードとして生成した鍵からメッセージから暗号文を生成

$Mac_K(M)$ ：K を鍵として，メッセージ M に対するメッセージ認証コードを生成

- パラメータ生成

セキュリティパラメータ1^λ，属性空間の次元の配列数 d，属性空間の次元 $\vec{n} = (n_1,\ n_2,\ \cdots,\ n_d)$ を入力とし，公開パラメータpk，マスター鍵mkを出力する．

$$Set\ up(1^\lambda, d, \vec{n}) \rightarrow pk,\ \ mk$$
$$pk := (\mathbb{B},\ g_T, 1^\lambda,\ \vec{n}, \mathrm{pub})$$
$$sk := \mathbb{B}^*$$

- 復号鍵生成

公開パラメータpkとマスター鍵mk，鍵識別子$\mathbb{S}$を入力とし，対応する秘密鍵$sk_{\mathbb{S}}$を出力する．

$$KeyGen(pk,\ mk,\ \mathbb{S}) \rightarrow sk_{\mathbb{S}}$$
$$sk_{\mathbb{S}} := (\mathbb{S},\ k^*_0,\ k^*_1,\ \cdots,\ k^*_l)$$

- 暗号化

公開パラメータpkと属性情報Γと平文mを入力とし，暗号文 $c_{t\Gamma}$を出力する．

$$Enc(pk,\ \Gamma,\ m) \rightarrow c_{t\Gamma}$$
$$(\Gamma := \{(t,\ \vec{x_t}) | 1 \le t \le d\})$$

以下のアルゴリズムからなる．

1. $(\mathbb{B},\ g_T, 1^\lambda,\ \vec{n}, pub) \leftarrow pk$
2. $\delta \xleftarrow{U} {\mathbb{F}_q}^3$
3. $\zeta,\ \ \varphi \xleftarrow{U} {\mathbb{F}_q}^{n_t}$
4. $(mackey, com, decom) \xleftarrow{U} S_{encap}(1^\lambda, pub)$
5. $c_t \leftarrow \left(\overbrace{\delta_1 \vec{x_t}}^{n_t},\ \overbrace{0^{n_t}}^{n_t},\ \overbrace{0^{n_t}}^{n_t},\ \overbrace{\varphi_t}^{1}\right)\mathbb{B}_t,\quad for\ (t,\ \vec{x_t}) \in \Gamma$
6. $c_{d+1} \leftarrow \left(\overbrace{\delta_1 - \delta_2}^{1},\ \overbrace{0}^{1},\ \overbrace{\zeta}^{1},\ \overbrace{0}^{1},\ \overbrace{\varphi}^{1}\right)\mathbb{B}_{d+1}$
7. $c_{d+2} \leftarrow \left(\overbrace{\delta_2 - \delta_3 \cdot com)}^{1},\ \overbrace{\delta_3}^{1},\ \overbrace{0}^{1},\ \overbrace{0}^{1},\ \overbrace{0}^{1},\ \overbrace{0}^{1},\ \overbrace{\varphi}^{1}\right)\mathbb{B}_{d+2}$
8. $\mathrm{m}' \leftarrow decom \parallel m$
9. $K \leftarrow g_T^\zeta$
10. $symkey \leftarrow KDF(K)$
11. $cm \leftarrow SE_{symkey}(m')$
12. $C \leftarrow (\Gamma,\ c_1,\ \cdots,\ c_d,\ c_{d+1},\ c_{d+2},\ cm)$
13. $mac \leftarrow Mac_{mackey}(C)$
14. $c_{t\Gamma} \leftarrow (com,\ C,\ mac)$

- 復号

公開パラメータpkと鍵識別子(述語) $\mathbb{S}$に対応する秘密鍵$sk_{\mathbb{S}}$と暗号文$c_{t\Gamma}$を入力として，正しく復号された場合には，復号文m'を出力する．

$$Dec(pk,\ sk_{\mathbb{S}},\ c_{t\Gamma}) \rightarrow m'$$

3. 提案方式

本章では，悪意のある第三者に対しセンサデータの漏洩を防ぎつつ，IoT 機器での鍵管理を容易にすることを目的として，暗号方式として関数型暗号を適用した IoT 機器間でのデータ授受方式を提案する．提案方式を構成するエンティティを説明した後，提案方式のシステム全体のシーケンスを述べる．また，提案方式を実現するに当たり，検討しなくてはならない課題を述べる．

3.1 提案方式の概要

提案方式のシステム概要を図 2 に示す．提案方式では，センサデータを送信する機器が事前に配布されたセンサ属性情報 Γ(例えば，”温度”という情報)を利用し，取得したセンサデータ $\mathrm{m_A}$ (例えば，”27 度”) を暗号化，IoT 機器 Z が事前に配布された復号条件式を含む復号鍵skを利用して，受信した暗号化されたセンサデータ $\mathrm{C_A}$ (以降，暗号化センサデータと呼ぶ．) の復号を行う．鍵配布センタは，所有者からの依頼を基に属性情報 Γ や復号条件を含む復号鍵skを生成する．

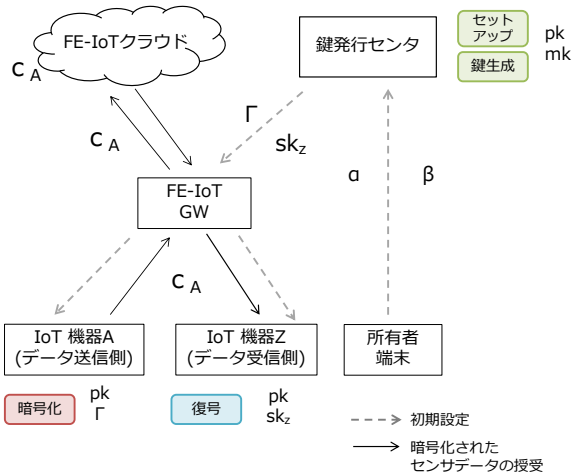

図 2　提案方式の概要

3.2 エンティティ

提案方式は，データを送信する IoT 機器 (以下，IoT 機器 A とする．)，データを受信する IoT 機器 (以下 IoT 機器 Z とする．)，関数型暗号の鍵発行を行う機能をもつ鍵発行センタ，IoT 機器へのセンサデータの閲覧許可を決定する IoT 機器の所有者端末，鍵発行センタと IoT 機器との通信

を補助する FE-IoTGW，暗号化されたセンサデータを格納する FE-IoT クラウドからなる. FE‐IoT GW 及び, FE‐IoT クラウドは，2.1 で述べた IoT GW，IoT クラウドに関数型暗号に対応する機能を新たに追加したものである．IoT 機器 A は, センサデータを暗号化して送付する側であり, IoT 機器 Z は，センサデータを活用する側である．各機器の機能詳細を以下に示す.

❑ FE-IoT GW

鍵発行センタと IoT 機器間，IoT クラウドと IoT 機器間の情報のやり取りの補助を行う.

鍵発行時に鍵発行センタから受信する鍵発行情報を保有しており，各 IoT 機器が閲覧を許可されているセンサデータの属性情報を把握している．センサデータの取得依頼が来た IoT 機器の ID を基に，IoT GW へ依頼元の IoT 機器が閲覧許可されている暗号化センサデータを要求する. IoT GW から受信した暗号化センサデータを IoT 機器へ送付する.

各 IoT 機器は特定のデータのみを閲覧可能な復号鍵のみを保有しているため，IoT 機器は自分が閲覧許可されたセンサデータのみを取得する必要がある．IoT GW が各 IoT 機器の閲覧許可されているセンサデータの属性情報を把握していることで，IoT 機器は自分が許可されたセンサデータを把握していなくても，自分の許可されたセンサデータを受信することができる．すべての IoT 機器に取得する情報の設定を行う方法も考えられるが，膨大な IoT 機器，IoT GW に対して一つ一つ設定を行う方法は現実的ではない.

❑ FE-IoT クラウド

IoT 機器が暗号化したセンサデータを保管する．IoT GW との暗号化センサデータの送受信を行う．IoT GW から送付された暗号化センサデータを属性情報と紐付けて保管する．各 IoT 機器が自身の取得するセンサデータの属性情報を用いてデータを暗号化して送信するため，さまざまな属性情報で暗号化された暗号化センサデータが IoT クラウド上に保管されている．IoT クラウドは，IoT GW から依頼されたセンサの属性情報に関わる暗号化データを IoT GW へ送付する.

❑ 鍵発行センタ

鍵発行センタは，マスター鍵，公開パラメータを保有している. 所有者端末からの依頼により, 属性情報 Γ の発行，復号鍵 sk の生成，発行を行う.

属性情報 Γ とは, 暗号化処理を行う IoT 機器 A に配布するセンサデータの属性情報のことを指す．復号鍵とは，復号処理を行う IoT 機器 Z に配布する閲覧許可された属性情報の条件式が含まれた鍵である.

鍵発行センタは，所有者から依頼された復号鍵生成依頼情報 β をもとに，復号条件式を生成する．復号条件式は，復号鍵生成依頼情報 β に記述された閲覧許可情報に記載されたセンサの属性情報を OR で連結させ生成する．生成した復号条件式，マスター鍵，公開パラメータをもとに，復号鍵skを生成する．生成した復号鍵を，β に記載されている機器へ送付する.

❑ IoT 機器 A (データ送信側)

IoT 機器(データ送信側)は, 機器が持っているセンサデータの属性情報 Γ と公開パラメータ pk を保有している．取得したセンサデータを暗号化し，IoT GW へ送付する.

❑ IoT 機器 Z (データ受信側)

IoT 機器(データ受信側)は, 鍵発行センタから配布された復号が許可されたセンサ属性情報から生成された復号鍵skを保有している．受信した暗号化センサデータを保有している復号鍵sk，公開パラメータpkを用いて復号し，センサデータを活用する．IoT 機器 A より，計算能力の高い機器も想定される.

❑ 所有者端末

所有者端末とは，IoT 機器を管理している所有者の端末のことである．例えば，スマートフォンやパソコンのような機器である．属性情報 Γ や復号鍵 sk_z の発行権限を持っており, Γ の生成情報である α や sk_z の発行依頼情報である β を鍵発行センタへ送付する.

3.3 前提条件

本提案方式を実現する上で，以下の 4 つの事項が前提条件となる.

前提 1：エンティティ間の認証が正しく行われること
前提 2：IoT 機器に配布した属性情報，復号鍵が第三者に漏洩しないこと
前提 3：IoT 機器に配布した属性情報，復号鍵を保持できること
前提 4：IoT 機器や IoTGW におけるログ情報の取得，保管ができること

前提 1 において, 認証を行う箇所としては IoT GW と IoT 機器間, 所有者端末と鍵発行センタ間, 鍵発行センタと IoT GW 間が挙げられる. IoT GW と IoT 機器間の認証では, IoT 機器及び IoTGW のなりすまし防止のために必要である．悪意のある第三者が IoT 機器の属性情報を抜き取り，IoT 機器のセンサデータを改ざんし，データを暗号化する可能性があるからである．同様に所有者端末と鍵発行センタ間でも所有者端末のなりすましを防ぐために，正しく認証する後，通信を行う必要がある．鍵発行センタと IoT GW 間

の認証としては，IoT GW がなりすまされ，想定していない IoT GW へ鍵情報，属性情報を配布しないために必要である．

前提 2 は，外部からの攻撃により IoT 機器の端末から属性情報や復号鍵が盗まれた場合，復号鍵を利用して想定していない相手にセンサデータの閲覧許可を与えてしまうこととなるため，必要となる条件である．このようなことを防ぐため，例えば IoT 機器には耐タンパ性の記憶領域を保有していることが必要である．

前提 3 については，IoT 機器において，揮発性メモリに情報を書き込むと電源 off 時には，電源 on 時に新たに記憶していた情報が揮発してしまう．そのため，どのように機器に秘密情報を保持しておくか，または IoT 機器の起動時に属性情報や復号鍵を配布するかの機器への初期設定が課題となる．

前提 4 は，不正な鍵発行やデータの暗号化が行われた際に，どこで行われたかを判断するために必要となる．悪意のある第三者による挙動解析のため，暗号化されたセンサデータの暗号化，復号，鍵の発効についてのログ情報を取得，保管しておく必要がある．

3.4 提案方式におけるシーケンス

本方式は大まかに以下の 4 つのステップからなる．

1. IoT 機器を識別する ID の登録
2. 属性情報，復号鍵の配布
3. センサデータの暗号化，送信
4. センサデータの復号，受信

ステップ 1 は，各 IoT 機器に鍵配布を行う際にどの機器へ送付するかを識別するために必要な処理である．ステップ 1 の流れを図 3 に示す．

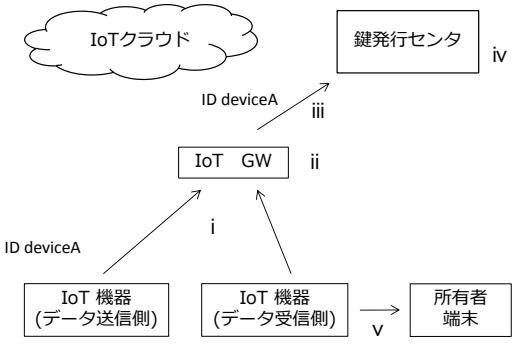

図 3　ID 登録

i. 各機器の固有 ID を IoTGW へ送付
ii. 取得した各機器の固有 ID を IoTGW の管理テーブルに登録
iii. IoTGW から鍵発行センタへ機器の固有 ID を登録
iv. 取得した各機器の ID を鍵発行センタの管理テーブルに登録
v. 各 IoT 機器から所有者端末へ ID 登録

ステップ 2 は，センサデータの暗号化を行うために必要な処理である．ステップ 2 の流れを図 4 に示す．

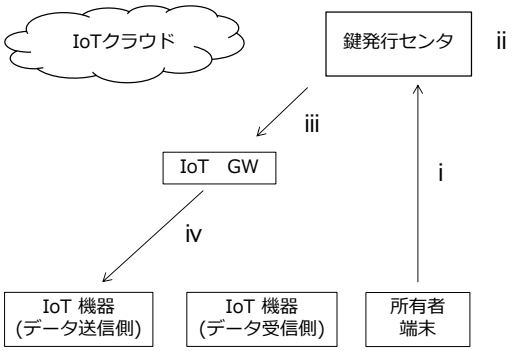

図 4　属性情報の配布

i. 所有者端末から属性発行依頼情報を送付
ii. 鍵発行センタにて取得した属性情報を登録
iii. 鍵発行センタから，IoTGW へ配布先機器情報，属性情報及び公開パラメータを送付
iv. IoTGW から IoT 機器へ属性情報及び公開パラメータを配布

ステップ 3 は，センサデータの復号を行うために必要な処理である．ステップ 3 の流れを図 5 に示す．

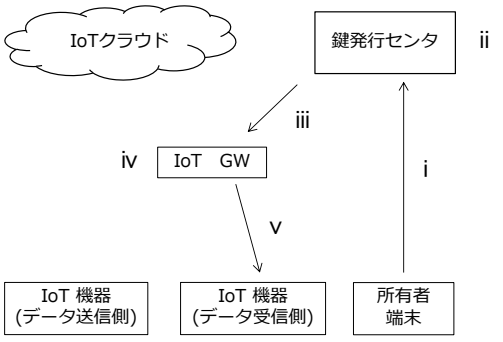

図 5　復号鍵の配布

i. 所有者端末から復号鍵発行情報を送付
ii. 鍵発行情報を基に，鍵発行センタにて復号鍵を生成
iii. 鍵発行センタから，IoTGW へ鍵生成情報，復号鍵及び公開パラメータを送付
iv. 取得した鍵生成情報を IoTGW のセンサ許可情報管理テーブルに登録
v. IoTGW から IoT 機器へ復号鍵及び公開パラメータを配布

ステップ 4，5 は，センサデータの暗号化，送信を行う処理である．ステップ 4，5 の流れを図 6 に示す．

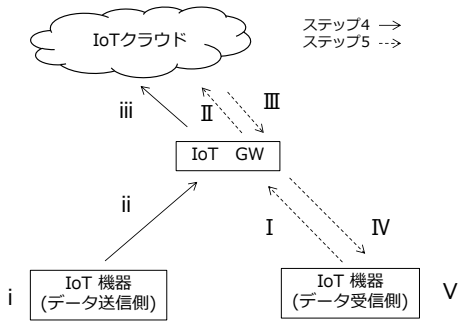

図 6　暗号化，復号処理

ステップ 4 についての処理の詳細は以下である．

i. IoT 機器にて，保有している属性情報，公開パラメータを用いて取得したセンサデータを暗号化
ii. 暗号化したセンサデータを IoTGW へ送付
iii. IoTGW から IoT クラウドへ暗号化されたセンサデータを送付し，保管

ステップ 5 についての処理の詳細は以下である．

I. IoT 機器にて，IoTGW へセンサデータの取得依頼
II. IoTGW の登録済みのセンサデータ許可管理テーブルから
該当するセンサデータを IoT クラウドへ取得依頼
III. IoT クラウドから IoTGW へ暗号化されたセンサデータを送付
IV. IoTGW から IoT 機器へ暗号化されたセンサデータを送付
V. 取得した暗号化センサデータを保有している復号鍵，公開パラメータにより復号

3.5 提案方式における関数型暗号に起因する課題

提案方式のように，IoT 機器システムに関数型暗号を適用したことによる課題としては，以下の 4 つが考えられるため，詳細に述べる．

課題 1：所有者とモノ(IoT 機器)の紐付け

IoT 機器を用いるシステムでは，所有者と IoT 機器が一対一で紐付いておらず，所有者が大量の IoT 機器を所有しているモデルが一般的である．しかし，これらを誰が所有者であるかの紐付けを鍵発行センタが理解していない場合，鍵発行センタと所有者の認証が正しく行われても，所有者以外から，鍵発行の依頼を行えてしまい，実際の所有者が意図していない復号鍵が発行される可能性が考えられる．そのため，鍵発行センタは鍵発行の依頼主が実際に発行を依頼している機器の所有者であるかを判定できる必要がある．

課題 2：配布した属性情報，復号鍵の失効，更新

属性情報を失効したい場合に一度配布した属性をどのように失効させるかという課題がある．これは，IoT 適用に限らず関数型暗号のアルゴリズムを用いるシステムでは常に発生する課題である．一般的な共通鍵暗号や公開鍵暗号でも失効というのは必要であるが，関数型暗号では暗号化と復号に利用する鍵が一対多であるため，より失効や更新の管理が煩雑になる．関数型暗号に関する，失効方式や再暗号化技術等については，研究が進められている[7][8].

課題 3：IoT 機器の識別

運用課題として，各機器へ異なる復号条件をもつ復号鍵を配布するため，IoT 機器の識別が必要である．各 IoT 機器を ID により識別する方法が一般的である．識別に用いる ID としては，各機器に出荷時に振られる製造番号や固有の ID を利用する方法があるが，これらは IoT 機器を出荷しているメーカなどに依存しており，ID が衝突する可能性もある．ID の衝突を回避する方法としては，IoT 機器で ID を生成する方法が考えられ，IoT 機器の製造時に発生する個体差を利用して固有の ID を生成する方式が検討されている[9][10].

課題 4：IoT 機器における暗号化，復号処理時間

性能課題としては，関数型暗号は共通鍵暗号や公開鍵暗号よりも高機能なことが実現できる反面，これらの暗号方式よりも計算量が大きいため，IoT 機器で処理を行えるのかという課題がある．

ここまで，本方式で考えうるシステム課題を網羅的に述べたが，以降本稿では，最も本提案方式を実現する上で重要と考えられる，性能に関する課題のみを取り上げることとする．

4. 性能評価

本章では，提案方式の有効性を評価するため，実現時に最も懸念される IoT 機器における関数型暗号の処理性能の評価を行った．

4.1 評価対象

本稿において，性能評価の対象としたデバイスについては，Raspberry Pi 3 を用いた．Raspberry Pi 3 の性能の詳細は表 1 に示す．表からも明らかなように，Raspbrry Pi 3 は，CPU コアがスマートフォンなどにも搭載されているような，ARM Cortex-A 系のデバイスである．現在，ARM コア内蔵チップはスマートフォンなどでも 90%以上の高いシェアを占めており[12], IoT 機器においても同様に高いシェアを占めることが予測される．また，今後 IoT 機器としてより安価な機器でも，Raspberry Pi3 相当の性能を持つ機器も増えていくと考えられる．特に今回想定しているような，センサデータを活用して IoT 機器自体の制御を行うような

機器を想定した場合には性能の高い機器が利用される可能性も高いと考え, 本稿では表 1 の機器を IoT 機器と想定し, 評価の対象とした. ただし, 4.4 考察でも述べるように ARM Cortex-M 系のデバイスも一部測定を行った.

表 1　測定デバイスの性能，環境

Raspberry Pi 3	
CPUコア	ARM Coretes-A53 (quad-core)
OS	Rasbian (version8.0)
Clock	1.2 GHz
RAM	1 GB
ROM	0

4.2 測定条件

今回は，Raspberry Pi 3 にて，関数型暗号の各処理における性能測定を行った.

関数型暗号は，利用する属性情報の数，復号条件式での AND や OR 条件の結合数によって, 暗号化や復号処理の時間は異なる．今回の提案方式は，Key Policy 方式であるため，復号鍵 *sk* に復号条件式が含まれている．関数型暗号の処理時間の測定では，いくつかの利用例を想定し，以下の表 2 に示す 4 つの条件について測定を行った．このとき，これらの条件について, 常に暗号化時の属性は *sk* の復号条件式を満たしているものとする．条件 2 は，3 章で説明した複数のセンサの種別を OR 条件で連結した復号条件式により生成した復号鍵 *sk* を利用する例である．3 章で述べた方式以外にも利用例としては，特定の日時のみでセンサデータを復号可能なように，センサの種別と時間情報を AND で連結させた復号条件式から生成した復号鍵 *sk* を利用する場合 (条件 3)や，更に特定のデバイスのみでセンサデータを復号可能なように，IoT 機器のデバイス ID，時間情報，センサ種別を AND 条件で連結させた復号条件式から生成した復号鍵 sk を利用する場合 (条件 4)の利用例が考えられる．条件 1 は，基準となる参考値として条件に含めた．これらの想定される利用例の条件式において関数型暗号の処理測定を行った.

表 2　関数型暗号の測定条件

	暗号化時の属性数	skの復号条件式
条件1	1	1属性
条件2	1	2属性のOR条件
条件3	2	2属性のAND条件
条件4	3	3属性のAND条件

4.3 測定結果

Raspberry Pi 3 における関数型暗号の各処理の測定結果を表 3 に示す．各処理時間は 10 回測定した平均値を示す.

表 3 Raspberry Pi 3 における関数型暗号の処理時間

処理		処理時間 / msec			
		条件1	条件2	条件3	条件4
	enc	587	586	741	897
	dec	568	569	764	967
(参考)	setup	4437	6108	5958	7451
	keygen	2002	2794	2794	3582

表 3 からわかるように，条件 1，条件 2 の復号時間に着目すると，復号鍵の生成時に用いる復号条件式に含まれる属性情報の数が増加しても，復号時間に影響がないことがわかる．一方，条件 3，条件 4 においては，復号時に必要な属性数が増加するほど，暗号化，復号処理ともに処理時間は増加している.

4.4 考察

4.3 の結果より，Raspberry Pi 3 において，関数型暗号を利用した暗号化の処理時間は，どの条件下においても，1 秒以下で処理が行えている．他の IoT 機器へのセンサデータの送付というユースケースを考えると，1 秒よりも短い間隔でセンサデータを送付し，機器の制御に利用するというシーンは考えにくい．よって，暗号処理において，Raspberry Pi 3 程度の性能を持っている機器であれば，関数型暗号は実用的な時間で利用できると言える．また，復号処理についても，1 秒以下で処理が行えている．暗号化時に利用する属性数が増加するほど，復号処理時間は増加しているが，利用例を想定した場合，暗号化時に 4 つ以上の属性を用いて復号する例は想定されないため，提案方式において復号処理時間は最大でも 1 秒と推測できる.

また，今回，Raspberry Pi 3 について，関数型暗号の測定を行ったが，センサデータを送信する側の IoT 機器としては ARM Coretex-A 系のデバイスである Raspberry Pi 3 よりも計算能力の低い ARM Coretex-M 系のデバイスの利用も想定される．これらのデバイスにおいては，送信するセンサデータの暗号化に関数型暗号を用いるため，暗号化処理の性能が重要となってくる．ARM Coretex-M 系のデバイスの性能例の詳細は表 5 に示す. これら ARM Coretex-M 系の IoT デバイスについては，RasberryPi のような高機能な OS の導入は難しく，たとえば mbedOS を組み込んだ環境での動作が必要となり，RaspberyPi で動作させたライブラリの単純移植は困難である．よって，関数型暗号の主な処理である，楕円スカラ倍算の演算性能を個々に測定することにより，実際に関数型暗号を実装した際の大まかな速度を見積もることとした．ARM Cortex-M 系のデバイスは FRDM-K64F， STM32F7Discovery を用いた. 各デバイスでの楕円スカラ倍算の処理の測定結果を表 4 に示す．処理時間は 100 回測定した平均値を示す．楕円曲線パラメータについては[11]を参考のこと.

表 4　楕円スカラ倍の処理時間

デバイス	楕円曲線パラメータ [10]	時間 msec/ecmul
Raspberry Pi 3	Fp254BN	9.7
	secp256k1	10.9
	secp256r1	10.2
	secp384r1	16.1
STM32 F7 Discovery	secp256k1	95
	secp256r1	121
	secp384r1	180
FRDM-K64F	secp256k1	180
	secp256r1	172
	secp384r1	202

2.2 に示しているように，関数型暗号の暗号化における処理の大部分は楕円スカラ倍算が占めている(表 3，4 の数値からも明らか)ため，楕円スカラ倍算の処理時間に着目する．今回実装した関数型暗号で利用されている楕円曲線は Fp254BN であるが，これは，256 bit の鍵長を持つ secp256k1 などと処理時間が近く，ARM Coretex-M 系のデバイスにおいても処理時間が近いと予測できる．よって，表 4 での Raspberry Pi 3 と ARM Coretex-M 系の 2 つのデバイスの処理時間から，楕円スカラ倍算の ARM Cortex-A 系のデバイスに比べ， ARM Cortex-A 系のデバイスでは関数型暗号の暗号化処理は 10〜15 倍程度時間がかかると推測できる．よって，表 3 での測定結果を参考にすると，ARM Cortex-M 系のデバイスでは，センサデータ送信時に暗号化処理として，推測で 10 秒程度かかるが，秒単位のリアルタイム性を求めないセンサデータであれば，分単位や時間単位で暗号化されたセンサデータを送付できればよく，そのような利用シーンにおいては，十分実用的であるといえる．

5. まとめ

本稿では，関数型暗号を適用し，IoT 機器のセンサデータの利活用を安全に実現できる方式を提案，また有効性について最も懸念される性能面から検討を行った．今回提案した方式は，IoT 機器のセンサデータの暗号化にセンサデータの属性情報を利用し，復号時には閲覧許可されたセンサの属性情報を含む条件式をもつ復号鍵によりセンサデータを復号する方式である．よって，煩雑な鍵管理を必要とせず，かつ許可しない悪意ある機器へはセンサデータを漏洩せずに，IoT 機器間の情報の授受が可能となる．これにより，IoT 機器同士が各々必要なセンサデータを活用し，膨大な機器の制御に活用することができる．また，Raspberry Pi 3 などの ARM Coretex-A 系のデバイスと同程度の性能の IoT 機器であれば，IoT 機器での暗号化，復号処理が 600msec 程度で実行できることが分かり，提案方式の有効性を示した．より計算能力が低いがセンサを収集する IoT 機器として利用される可能性のある ARM Coretex-M 系のデバイスに関しては，関数型暗号の暗号化処理の大部分を占める楕円スカラ倍算の測定結果からの見積もりより，ARM Coretex-A 系のデバイスと比較し，暗号化処理に 10 倍程度かかることが推測できた．今後の課題としては，ARM Coretex-M 系のデバイス上での関数型暗号の実測による消費電力やメモリ消費に関する検討が挙げられる．また，より実用的にするため，IoT 機器での関数型暗号の処理の高速化が挙げられる．今回の測定は事前計算テーブルを利用せずに演算を行ったが，事前計算を行い各処理を行うことにより，処理の高速化が実現できる可能性もある．また， IoT GW など他の機器へ一部の処理を委託することも検討していく必要がある．

参考文献

[1] “IoT デバイス数は 2017 年，世界人口を上回る？--ガートナー”，https://japan.zdnet.com/article/35096854/

[2] 坂村健，“IoT 動く”，http://www.tronshow.org/2016-tron-symposium/session-pdf/ja/pdf/20161214-1-1.pdf ，2016 TRON Symposiom.

[3] IPA,“IoT 開発におけるセキュリティ設計の手引き”, p. 18-19.

[4] T. Okamoto，K. Takashima，“Fully Secure functional Encryption with General Relations from the Decisional Linear Assumption，” CRYPTO2010，LNCS6223，pp. 191-208，2010. .

[5] 高島克幸, 岡本龍明,“アクセス権限を制御できる最新技術「関数型暗号」,” 日経 BP 社，p. 87-95，2012.

[6] Boneh, D.,Katz, J.,Impruved efficiency for CCA-secure cryptosystems build using identity based encryption, In: Menezes, A. (ed.) CT-RSA 2005, LNCS, vol.3376, pp.87-103, Springer, Heidelgerg(2005)

[7] KDDI，“世界初！加速度センサーの個体差により固有の ID を生成する技術を開発”，http://internet.watch.impress.co.jp/docs/news/1024467.html，2016.

[8] 三菱電機,“LSI の個体差から指紋のような固有 ID を生成し，組み込み機器の安心・安全に貢献「IoT 時代見向けたセキュリティー技術」”，http://www.mitsubishielectric.co.jp/news/2015/0205.pdf ，2016.

[9] 河合豊, 高島克幸,“代理人再暗号化機能を持つ関数型暗号”, SCIS2014.

[10] 伊藤隆，市川幸広，森拓海，河合豊，高島克幸，“関数型暗号における失効管理方式に関する考察”，SCIS2014.

[11] Standard of Efficient Cryptograpy Group，“SEC 2：Recommended Elliptic Curve Domain Parameters，1999.

[12] ”明確になったソフトバンク買収後の ARM の IoT 戦略”, http://pc.watch.impress.co.jp/docs/news/1033138.html

表 5　評価対象のデバイスの性能

開発ボード	CPUコア	Clock	RAM	ROM	wifi	BLE	Ethernet
Raspberry Pi 3	ARM Coretes-A53	1.2 GHz	1 GB	0	○	○	○
FRDM-K64F	ARM Coretex-M4F	120 MHz	256 KB	1024 KB	○	○	○
STM32 F7 Discovery	ARM Coretex-M7	216 MHz	340 KB	1 MB	×	×	○
Discovery kit with STM32F429ZI MCU	ARM Coretex-M4	180 MHz	256 KB	2 MB	×	×	×
Arduino M0 Pro	ARM Coretex-M0+	48 MHz	32 KB	256 KB	×	×	×