準同型暗号によるセキュア連想メモリ

辺 松[1,a)]　廣本 正之[1]　佐藤 高史[1]

概要：本論文では準同型暗号を用いたセキュアな連想メモリを提案する．連想メモリに格納されるデータを準同型暗号によって暗号化し，暗号文のままデータワードのマッチングを行うことでセキュアな検索を実現する．アルゴリズム面では，ワードマッチングを加法準同型暗号のみを用いて実装することを提案する．提案手法では，乗法準同型が必要になる XNOR-AND によるマッチングではなく，XOR-OR でワードマッチングを計算することで，暗号文同士の簡単な加算でワードマッチングを行う．またアルゴリズム面での最適化により，簡単かつ並列性の高いハードウェアアーキテクチャで，効率的なセキュア連想メモリを実装することを可能にする．数値実験により，提案手法のソフトウェア実装が既存手法より 403 倍高速化であることを示した．また専用ハードウェアの実装により，消費エネルギーを 477 万倍削減できることを示した．

Secured Content Addressable Memory Based on Homomorphic Encryption

Song Bian[1,a)]　Masayuki Hiromoto[1]　Takashi Sato[1]

Abstract: We propose an implementation of a secured content addressable memory (SCAM) based on homomorphic encryption (HE), where HE is used to compute the word matching function without the processor knowing what is being searched and the result of matching. By exploiting the shallow logic structure (XNOR followed by AND) of content addressable memory (CAM), we show that SCAM can be implemented with only additive homomorphism, greatly improving the efficiency of the HE algorithm. In the proposed method, the logic of homomorphic XNOR-AND is replaced with homomorphic XOR-OR, requiring only simple additions to be performed on the ciphertext. We also show that our scheme can be implemented by highly parallelizable and simple hardware architecture. Through experiment, we demonstrate that our software implementation is already 403x faster than the fastest known algorithm. With the help of hardware, we can achieve an energy reduction per word match by a factor of 477 million times, making our SCAM scheme much more practical.

1. 序論

近年，主要企業がその内部情報をクラウドサービスにアウトソーシングする機会が増加しており [1]，プライバシーを考慮した情報検索 (Private Information Retrieval; PIR) が重要な研究分野となっている．PIR は，ユーザがサーバから必要な情報を読み出す際に，それが何であったかをサーバに知られずに情報取得を行う手法のことを指す．このような PIR を実現する手法の中で，準同型暗号 (Homomorphic Encryption; HE) が理論的に良い選択肢とされており [2]，多数の手法が提案されている [3–5]．しかし，一般にこれらの PIR のクエリは複雑であり，かつ，任意のサイズを持つデータベースに適用する必要がある．このため，巨大なデータベースレコードそれぞれに対する複雑な準同型関数の計算が必要となり，また，十分なセキュリティを確保するために必要となる準同型暗号のパラメータ数も増大するという問題がある．よって現状では準同型暗号による PIR の実現は現実的と言えない状態である．例として，10 個のレコードを持つデータベースに対するクエリ処理に 7 日間を要するという報告もある [5]．

より高速に PIR を行うには，連想メモリ (Content Ad-

[1] 京都大学 大学院情報学研究科 通信情報システム専攻
〒 606-8501 京都府京都市左京区吉田本町
[a)] paper@easter.kuee.kyoto-u.ac.jp

dressable Memory; CAM) の利用が有用である．CAM とは，単一クロックサイクルで全てのメモリコンテンツに対し並列の検索を行うことが可能なメモリである [6]．CAM はパケット通信 [7,8] やデータベース [9] における検索の高速化等に広く利用されている．一般に二値 CAM (Binary CAM; BCAM) のセルは 9 個のトランジスタによって実現でき，非常に効率的であることが知られている [10]．ただしこの効率的な実現は，クエリ自体が平文に対する検索であることが前提であり，プライバシーについては全く考慮されてない．

プライバシー保護の観点から，安全な CAM (Secured CAM; SCAM) の設計は重要な課題である．すなわち，ユーザが SCAM に対してデータを送り，サーバはそれに値が一致するアドレスを返す．ここで，ユーザの送ってきたデータの内容をサーバが知ることのできない CAM が設計できれば，ユーザのプライバシーは保証される．この SCAM は，ある意味，簡略化された PIR スキームであると言える．SCAM では，複雑なクエリではなく，単純に検索したいデータをサーバに送り，それに対応するアドレスを手に入れるだけである．具体的には，ベクタ $\mathbf{x}$, $\mathbf{y}$ の各要素 $x_i, y_i \in \{0,1\}$ が，数字 x, y の i 番目のビットを表す場合，従来の BCAM は以下の関数を計算する．

$$f(\mathbf{x},\mathbf{y}) = \prod_{i=1}^{w} \overline{x_i \oplus y_i} \qquad (1)$$

ここで，w は x と y のビット長を意味する．この関数のみを準同型暗号によって実現する場合，より高速な計算が可能であることが Khedr らによって示されている [11]．しかし，後述のように，Khedr らは単純に GSW と呼ばれるスキームを用いて式 (1) を実現しているため，準同型乗算などの非常に遅い計算の実行を強いられており， 計算速度の低下を招いている．

本論文では，新たなセキュア連想メモリ SCAM と，そのハードウェア実装を提案する．提案 SCAM は，安全な BCAM の実現を目標としている．ワードマッチングの計算が非常に浅い論理回路によって実現できることを利用して，新しい準同型暗号による安全なワードマッチング方法を設計する．提案暗号方式は既存研究 [12] に基づくが，それを改変しハードウェア実装を意識した暗号方式を提案する．具体的には，既存研究である単純なアクセラレータ [13] や，従来のブートストラッピングを用いる準同型暗号方式 [12]，Khedr らによる GSW 実装 [11] と異なり，提案手法では倍精度乗算や FFT 演算等をアルゴリズム上で不要とし，効率的なハードウェア実装を可能にした．その結果，ソフトウェア実装では既存研究の 403 倍の高速化を実現し，さらに専用ハードウェア化により既存手法 [11] に対し 8 桁の電力削減を達成できることを示した．本論文の貢献を以下にまとめる．

- **準同型暗号の新しい応用例の提案**: 準同型暗号による安全な連想メモリの実装を初めて提案する．連想メモリは広く使われているが，その安全な実装は我々の知る限りではこれまで提案されていない．
- **ハードウェア向け準同型暗号スキーム**: FHEW [12] を元にした XOR-OR 準同型ゲートによるマルチビットワードマッチ関数を提案する．既存研究である FHEW は，単一ビットに対する NAND ゲートしか準同型に計算できないが，提案手法では FHEW の暗号方式自体を変えることでより複雑な複合演算を可能にした．
- **SCAM 向けハードウェア設計**: 提案手法のハードウェアアーキテクチャを検討する．準同型計算の複雑性により，要求されるハードウェア資源は暗号化を行わない CAM に比べて数桁増加する．例えば，平文での単一ビット OR は 16 KB の暗号文の加算に相当する．よって，SCAM 向けハードウェアのアーキテクチャ探索について，本問題固有のトレードオフ検討が必要となる．

2. 準備

2.1 表記

本論文では $\mathbb{Z}_2^w$ 上のベクタを $\mathbf{x}$, $\mathbf{y}$ で表わす．ここで w はビット長である．すなわち $\mathbf{x}$，$\mathbf{y}$ はそれぞれ w ビットの数 x，y の，二進数表現であるとみなせる．

N は CAM のライン数を表し，各ラインは w ビットの平文整数を保持するものとする．また，$\lg x$ は $\log_2 x$ を表す．

セキュリティの表記においては，m が平文，c が暗号文，s が秘密鍵を，それぞれ表す．Enc と Dec はそれぞれ暗号関数と復号関数である．(q, n, χ_σ) は準同型暗号のパラメータ群であり，それぞれ，整数環 Z_q，次元数 n，および σ のパラメータを持つ分布 χ を表す．

2.2 LWE 問題と準同型暗号

整数 LWE (Learning With Errors) 問題は，Regev [14] および Paikert [15] により初めて提唱された困難性問題である，最悪ケースの格子に対する近似最短ベクトル問題へ，量子的および古典的に帰着できることが示されている．LWE 問題の簡潔さと自由度の高さにより，多数の暗号システムが LWE 問題を元にしている [16–18]．

本研究では，Ducas と Micciancio [12] により改変された一般的な LWE に基づく準同型暗号 (FHEW) を使用する．FHEW の構成としては，まず要求セキュリティレベル λ に応じたパラメータ (q, n, χ_σ) を決定する．一般的な LWE では，χ_σ は標準偏差 σ を持つ離散ガウシアン分布を表す．詳しい定義は [14] の式 (6) を参照されたい．整数 $t < q$ があるとし，メッセージ m はメッセージ空間 $\mathbb{Z} \bmod q$ から選ぶことができる．メッセージを暗号化するには，まず二つのベクタ $\mathbf{a}, \mathbf{s} \in_R \mathbb{Z}_q^n$ をサンプリングする．ここで $\in_R \mathbb{Z}_q^n$ は一様に $\mathbb{Z}_q$ からランダムに n 回サンプリングすることを

表 1　NAND ゲートの真理値表

m_0	m_1	$\frac{5}{4} - \frac{m_0}{2} - \frac{m_1}{2} \pm 4\varepsilon$
0	0	$\lfloor \frac{5}{4} \pm 4\varepsilon \rceil = 1$
0	1	$\lfloor \frac{3}{4} \pm 4\varepsilon \rceil = 1$
1	0	$\lfloor \frac{3}{4} \pm 4\varepsilon \rceil = 1$
1	1	$\lfloor \frac{1}{4} \pm 4\varepsilon \rceil = 0$

意味する．次に $\varepsilon \leftarrow \chi_\sigma$ を分布 χ_σ からサンプリングする．ここで ε は「小さい」整数とする (q と比較して $\varepsilon \ll q$ である)．そして暗号関数 Enc とそれに対応する復号関数 Dec は以下のように計算できる．

$$b = Enc(m) = \langle \mathbf{a}, \mathbf{s} \rangle + \frac{mq}{t} + \varepsilon \bmod q \in \mathbb{Z}_q \tag{2}$$

$$m = Dec(b) = \left\lfloor \frac{t}{q}((b - \langle \mathbf{a}, \mathbf{s} \rangle) \bmod q) \right\rceil \bmod t \in \mathbb{Z}_t \tag{3}$$

ここで q, n, $\mathbf{a}$ は公開され，暗号文に対する準同型関数の実行を可能にする．暗号文はこの場合，$\mathbf{c} = (\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$ で記述する．暗号文は $|\varepsilon| < q/2t$ である限り復号できる．加法準同型を実現する場合は，この誤差を $|\varepsilon| < q/4t$ とする．するとこの条件を満足する二つの暗号文を足しても誤差は $q/2t$ より小さくなり，復号可能であることが分かる．この暗号システムにおける乗法準同型の実現はコストが大きいことが知られており，代表的な手法としてテンソル積 [17] や再線形化とモジュロ切替を組み合わせた手法 [18, 19] が存在する．

乗法準同型の計算量が大き過ぎることから，乗法準同型を用いない NAND ゲートの実現方法が Ducas と Micciancio により以下のように示されている．例えば平文空間 $\mathbb{Z}_4$ 上の 2 つのメッセージ m_0 と m_1 をそれぞれ $c_0 = (\mathbf{a}_0, b_0)$ と $c_1 = (\mathbf{a}_1, b_1)$ として暗号化すると，誤差は $|\varepsilon| < q/4t = q/16$ となる．この時，式 (4) および表 1 から，異なる平文空間 $\mathbb{Z}_2$ における NAND(m_0, m_1) の暗号は，$\left(-\mathbf{a}_0 - \mathbf{a}_1, \frac{5q}{8} - b_0 - b_1\right)$ によって正しく与えられる．

$$\begin{aligned} &Dec((-\mathbf{a}_0 - \mathbf{a}_1, \frac{5q}{8} - b_0 - b_1)) \\ &= \left\lfloor \frac{2}{q} \left(\frac{5q}{8} - (b_0 + b_1) - (\mathbf{a}_0\mathbf{s} + \mathbf{a}_1\mathbf{s}) \right) \right\rceil \bmod 2 \\ &= \left\lfloor \frac{2}{q} \left(\frac{5q}{8} - \frac{m_0 q}{4} - \frac{m_1 q}{4} \pm \varepsilon_0 \pm \varepsilon_1 \right) \right\rceil \bmod 2 \end{aligned}$$

全ての $|\varepsilon_i| < q/16$ をひとつの ε にまとめると，

$$\left\lfloor \frac{5}{4} - \frac{m_0}{2} - \frac{m_1}{2} \pm 4\varepsilon \right\rceil \bmod 2 \tag{4}$$

が得られる．

3. 準同型暗号によるセキュア連想メモリ

3.1 2 段階 SCAM プロトコル

連想メモリの振舞いは PIR とは若干異なる．一般的な準同型による PIR では，暗号化されたアドレスがサーバに送られ，そのアドレスに対応するデータをユーザに送り返す [20]．一方，CAM はデータが入力で，そのデータワードと合致する CAM ラインのアドレスが出力となる．また，安全な CAM ではこのアドレスをサーバに知られてはならない．よって我々は図 1 に示す 2 段階の通信プロトコルを採用する．ユーザは暗号化されたクエリ c_{request} をサーバに送り，サーバは応答としてマッチング結果 $c_{\text{match}_1}, c_{\text{match}_2}, \cdots, c_{\text{match}_N}$ を送り返す．ここで N はデータベースのサイズ，つまり CAM のライン数である．

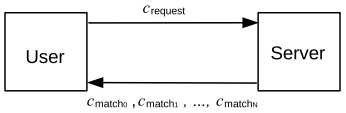

図 1　2 段階通信プロトコル

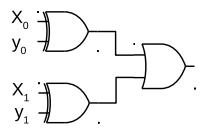

図 2　2 ビット XOR-OR

表 2　2 ビット XOR-OR 複合ゲートの真理値表

x_0	y_0	x_1	y_1	XOR-OR	$\widehat{\text{HomXOR-OR}}$
0	0	0	0	0	0
0	0	0	1	1	−1
0	0	1	0	1	1
0	0	1	1	0	0
0	1	0	0	1	−2
0	1	0	1	1	−3
0	1	1	0	1	−1
0	1	1	1	1	−2
1	0	0	0	1	2
1	0	0	1	1	1
1	0	1	0	1	3
1	0	1	1	1	2
1	1	0	0	0	0
1	1	0	1	1	−1
1	1	1	0	1	1
1	1	1	1	0	0

3.2 提案準同型暗号スキーム

式 (1) の実現において，XNOR と AND ゲートそれぞれは FHEW を用いて実現できる．しかしそれらの組合せである XNOR-AND 複合ゲートの実現は理論上困難である．そこで代替案として，XOR-OR 複合ゲートの実現を考える．XNOR-AND の結果は，XOR-OR の否定で得られる．簡単な 2 ビット XOR-OR ゲートの例を図 2 に示す．

準同型 XOR-OR ゲートの設計において加法準同型のみを用いる場合には，以下のような「キャンセリング問題」が発生する．準同型 XOR ゲートを正確に計算すると $\text{HomXOR}(\mathbf{c}_{x_i}, \mathbf{c}_{y_i}) = (\mathbf{c}_{x_i} - \mathbf{c}_{y_i})^2$ となり，乗法準同型が必要となる．ここで $\mathbf{c}_{x_i}$ と $\mathbf{c}_{y_i}$ はそれぞれ x_i と $y_i \in \mathbb{Z}_2$ の暗号文を意味する．ここで，乗算を用いない擬似的な方法として，

$$\widehat{\text{HomXOR-OR}}(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{w} (\mathbf{c}_{x_i} - \mathbf{c}_{y_i}) \tag{5}$$

を考えることができる．しかし式 (5) の計算結果には非対称性がある．例えば，$x_i = 1, y_i = 0$ の時に，$\widehat{\text{HomXOR}}$ は 1，$x_i = 0, y_i = 1$ の時に -1 になる．この場合，二つの $\widehat{\text{HomXOR}}$ の結果を図 2 の計算により足し合わせると，1 と -1 は打ち消し合って 0 となり（キャンセリング），結果はミスマッチではなくマッチとなってしまう．

キャンセリング問題を解決するため，$\mathbf{x}$ および $\mathbf{y}$ の i ビット目を暗号化する過程において暗号化定数 k_i を導入する．ここで $k_i = 2^i$ とし，この k_i を許容するために平文空間を $\mathbb{Z}_4$ から $\mathbb{Z}_t$ に拡張し，$t = k_{w+1} = 2^{w+1}$ とする．$\mathbf{x}$ および $\mathbf{y}$ の各ビットは暗号化定数と一緒に暗号化される．このとき，暗号および復号関数は以下のようになる．

$\mathbf{x} = [x_w, x_{w-1}, \cdots, x_0]^{\mathrm{T}}$ の各要素 $x_i \in \{0, 1\}$ に対し，

$$c_{x_i} = Enc(x_i) = (\mathbf{a}, b),$$
$$\text{where} \quad b = \langle \mathbf{a}, \mathbf{s} \rangle + \frac{k_i x_i q}{t} + \varepsilon \bmod q \tag{6}$$
$$k_i x_i = Dec(c_{x_i}) = \left\lfloor \frac{t}{q}((b - \langle \mathbf{a}, \mathbf{s} \rangle) \bmod q) \right\rceil \bmod t \tag{7}$$

ここで注意すべきは，復号関数の結果は x_i ではなく $k_i x_i$ となるため，マッチ時にゼロ，非マッチ時にゼロ以外の数値を取ることになる．

3.2.1 正当性

補題 1. 式 *(6)* と *(7)* の暗号，復号関数を用いれば $\mathbf{x}$ と $\mathbf{y}$ の相等関係は準同型的に計算できる．

証明. 補題 1 を帰納法で証明する．

基底段階: $w = 2$ の場合, 2 つの 2 ビット数 $\mathbf{x} = (x_1, x_0)$ と $\mathbf{y} = (y_1, y_0)$ がある．式 (5) を分解すると，

$$\widehat{\text{HomXOR-OR}}(\mathbf{x}, \mathbf{y}) = c_{x_1} - c_{y_1} + c_{x_0} - c_{y_0}$$

が得られる．暗号関数自体は線形関数であることから，メッセージ項とその他の *others* 項を分離することができる．

$$\frac{q}{t}(2x_1 - 2y_1 + x_0 - y_0) + others \tag{8}$$

others 項は復号によって消える．残った項の値を表 2 に示す．結果がゼロであるかどうかのテストを行うことにより，マッチ判定を行えることが分かる．

帰納段階: $\widehat{\text{HomXOR-OR}}(\mathbf{x}, \mathbf{y})$ は $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^{w-1}$ の相等関係を正しく計算できるとする．w 番目のビット を $\mathbf{x}$ と $\mathbf{y}$ に加えると，以下の式が得られる．

$$\widehat{\text{HomXOR-OR}}((x_w, \mathbf{x}), (y_w, \mathbf{y})) = \mathbf{c}_{x_w} - \mathbf{c}_{y_w} + \sum_{i=0}^{w-1}(\mathbf{c}_{x_i} - \mathbf{c}_{y_i})$$

ここで $\sum_{i=0}^{w-1}(\mathbf{c}_{x_i} - \mathbf{c}_{y_i})$ は $\mathbf{x}$ と $\mathbf{y}$ がマッチする時のみ 0 に復号される．基底段階と同様に項を分離すると，

$$\frac{q}{t}\left(2^w(x_w - y_w) + \sum_{i=0}^{w-1} 2^i(x_i - y_i)\right) + others \tag{9}$$

が得られる．ここで，i) $\mathbf{x}$ と $\mathbf{y}$ がマッチする場合と，ii) $\mathbf{x}$ と $\mathbf{y}$ がミスマッチする場合の 2 つの場合がある．i) では，x_w と y_w のマッチ結果によって全体の結果が決まる．$x_w = y_w$ の時は式 (9) は 0 となり，そうでない場合は 2^w か -2^w となる．従って i) の場合，式 (9) は正しい．ii) の場合，式 (9) の和の項の範囲は $S \in [-(2^w - 1), -1] \cup [1, (2^w - 1)]$ であることから，計算結果は必ずミスマッチ，つまり非ゼロとなる．つまり，x_w と y_w がマッチする場合，和の結果によりミスマッチが得られる．x_w と y_w がミスマッチする場合，式 (9) の結果は $\pm 2^w + S$ であり，ゼロになることはない．以上により，式 (9) は正しい計算結果を与えることが示された． □

3.2.2 セキュリティ

本論文では，平文空間が $t = 4$ から $t = 2^{w+1}$ と非常に広くなっているが，暗号と復号のスキーム自体は変えてはいないため，FHEW とは非常によく似たスキームである．よって，セキュリティの証明は [12,21] と同様に得られる．LWE に基づく暗号方式に対する攻撃手法のうち，現在知られている最良のものの計算時間は以下の root-Hermite factor δ に依存することが知られている [21].

$$\delta = 2^{(\lg^2 \beta)/(4n \lg q)}, \quad \text{where } \beta = (q/\sigma) \cdot \sqrt{\ln(1/\varepsilon)/\pi} \tag{10}$$

ここで $\ln x$ は自然対数を表す．平文空間の大きさを確保するためには，t より十分に大きい q が必要となる．q が大きいと，次元 n を大きくしなければ攻撃が容易になる．従って，平文空間を許容できる q に対し，その大きさにより n を調整しつつセキュリティの保証を行う必要がある．具体的なパラメータは 4 章で示す．

3.3 ハードウェアアーキテクチャ

提案する HE スキームは，高い並列性を持つことに加え，加算器のみを使用するため，ASIC による実現が非常に効率的であると考えられる．提案 SCAM のハードウェアアーキテクチャの全体像を図 3 に示す．ユーザは w ビットの平文に相当する $n \cdot \lg q$ ビットの暗号文をサーバに送る．サーバは，w 個の SCAM セルからなる SCAM ラインにより，N 個のワードに対するマッチングを行う．各 SCAM ラインでは，SCAM セルにより準同型的にビットごとのマッチングを行い，その結果に対し $w \cdot \lg q$ ビットの加算器木で OR を求め，ラインごとのマッチング結果を算出する．この結果は図 1 のようにユーザに送り返される．

SCAM セルの内部構造を図 4 に示す．SCAM セルは，$n \cdot \lg q$ ビットのメモリセルと，$\lg q$ ビットレジスタ，および $\lg q$ ビット加算器から構成される．暗号文は $\mathbb{Z}_q$ 上の整数からなるベクタであることから，加算器は $\lg q$ ビットの幅が必要である．q を 2 の累乗数にすることで，Z_q におけるモジュラ計算は単純に桁溢れを無視した $\lg q$ ビットの加

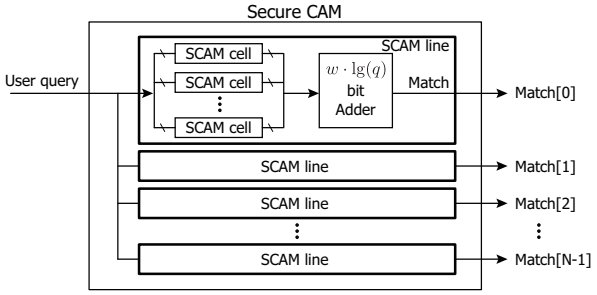

図 3 提案 SCAM のハードウェアアーキテクチャ

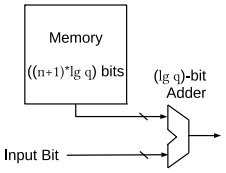

図 4 SCAM セルの内部構造

表 3 提案手法のパラメータ

$\lg t$	$\lg q$	n	σ	Adv. $\lg \varepsilon$	δ	Security (λ)
33	42	1052	8	64	1.00658	80 (Medium)
33	42	1318	8	64	1.00525	128 (High)

算になる．メモリは 1 ビットの平文に対応する暗号文を保持する必要があるため，$(n+1)\cdot\lg q$ ビットのサイズが必要である．

各 SCAM セルは $n+1$ 次元の $\lg q$ ビット整数を持つベクタを生成する，よってこれらを OR で一つの暗号文にまとめるには $\lg q$ ビットの加算器木が必要である．ここでは回路規模を抑えるため，$n+1$ 次元の足し算を逐次的に計算する．なお，十分な回路資源が利用可能な場合は，これらの加算を完全並列に行うことにより，準同型ビットマッチングを単一クロックサイクルで行うことも可能である．

4. 評価

4.1 LWE パラメータとストレージ使用量の改善

80 ビットおよび 128 ビットの 2 種類のセキュリティレベルを想定したパラメータ設定例を表 3 に示す．なお各パラメータは文献 [21] の式 (5.2) により計算した．32 ビット整数の暗号化を許容するために，平文空間は $t = 2^{33}$ に設定する．これにより，単一ビットの平文は，80 ビットのセキュリティで 5.523 KB，128 ビットのセキュリティで 6.919 KB の暗号文にそれぞれ対応する．これは，既存手法である SHIELD [11] と比べ，80 ビットセキュリティで暗号文のサイズを 1/83 に削減できたことを示している．また，提案手法の暗号文サイズは w に左右されるため，2 ビットのみの比較であれば 128 ビットセキュリティでも暗号文を 440 バイトに収めることができる．

4.2 CPU 実装

提案 SCAM の CPU 実装は，FHEW [12] の公開ソースコードを元に，ブートストラッピングの部分を削ったものとして実装した．プログラムは C++で記述し，実験は Intel Xeon E5-2630 2.3 GHz プロセッサと 32 GB のメモリを搭載した PC 上で行った．

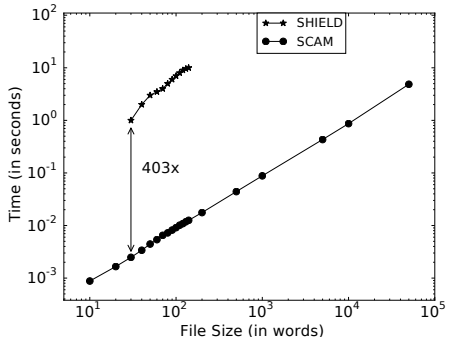

図 5 提案手法と SHIELD [11] の実行時間（80 ビットセキュリティ）

理論上，CPU による $\widehat{\text{HomXOR-OR}}$ の実装では，まず，暗号空間における準同型加算が $n+1$ 回の整数加算と AND 計算（モジュロ q）により計算される．これがビット数分だけ（w 回）実行され，計算された w 個の暗号文を一つに足し合わせる過程で $w-1$ 回の準同型加算が行われる．整数加算と整数 AND は単一 CPU サイクルで計算できると想定すると，一行の SCAM 結果を計算するのに $(2\cdot w-1)\cdot 2(n+1)$ クロックサイクル必要になる．実際の数字で計算すると，80 ビットセキュリティで $(2\cdot 32-1)\cdot 2\cdot(1052+1) = 132,678$ サイクルが必要になる．よって計算上では，CPU では 57.7 μs につき一行の SCAM ラインの比較を行うことができ，スループットは毎秒 17,331 ラインとなる．以上の計算はすべて 80 ビットセキュリティに基づいたものである．

実機での実測結果を図 5 に示す．前述の理論値よりはやや劣るが，80 ビットセキュリティで 91 μs で SCAM ラインの比較が行えている．SHIELD と比べると，時間複雑性を $O(n^{2.3727})$（行列乗算）から $O(n)$（行列加算）に，空間複雑性を $O(n^2)$ から $O(n)$ に，それぞれ削減した．その結果，各ワードが 32 ビット数だと想定した場合，30 ワードの検索を 403 倍高速化でき，12 コア並列では 4,836 倍高速化可能である．この比較は一般的な CPU と高性能 GPU との間の比較なので，提案手法は既存手法よりはるかに高速であると言える．

4.3 ハードウェア実装

本節では，SCAM の CPU 実装と ASIC 実装を比較する．提案手法により計算が非常に簡単になったことから，ASIC による SCAM の実装は電力を大幅に抑えることができる．図 3 と 4 に示すアーキテクチャを Verilogy HDL により実装し，商用 65 nm プロセスで論理合成を行った．単一 SCAM ラインを実装に必要な電力，面積，遅延時間を表 4 に示す．回路面積は 2 入力 NAND ゲート数への換算値である．$\widehat{\text{HomXOR}}$ と $\widehat{\text{HomOR}}$ の計算は 2 つのパイプライ

表 4　単一 SCAM ラインの論理合成結果

消費電力	回路面積（ゲート数換算）	遅延時間
1.205 mW	52,198	9 ns

表 5　各手法の性能比較

	SHIELD [11]	SCAM (CPU)	SCAM (ASIC)
暗号文サイズ	487.5 KB	6.8 KB	6.8 KB
計算時間	0.033 s	7.58 μs	9.47 μs
電力	165 W	95 W	1.205 mW
エネルギー	5.445 J	0.72 mJ	11.41 nJ

ンステージで並列化できるため，単一 SCAM ラインの比較は $n+1$ サイクルで行える．80 ビットセキュリティでは，これは 9.47 μs に相当する．計算速度の面では，CPU と比べ ASIC は 10 倍の高速化しか達成していないが，電力面では，CPU の 95 W の平均電力と比べて ASIC では 2.4 mW に抑えることができ，約 1/40,000 の電力削減を達成した．

全体の性能比較を表 5 に示す．ここで CPU は 12 コア並列であり，ASIC は単一 SCAM ラインの実装値である．計算時間は 1 ワードの比較に要する時間である．提案 SCAM の ASIC 実装により，消費エネルギーを SHIELD と比較して 8 桁削減することができた．

5. 結論

新たな準同型暗号を用いた SCAM スキームの提案とその評価を，ソフトウェアとハードウェアの両方で行った．乗法準同型やブートストラッピングの必要性を排除し，ハードウェアによる効率的な実装が可能なマッチングアルゴリズムを提案した．評価実験により，最速の既存手法よりも 4,836 倍早く計算でき，また ASIC 化を行うことで，さらに 40,000 倍の電力削減も可能であることを示した．

謝辞　本研究の一部は JSPS 科研費 26280014, 17H01713, 15K15960 および JSPS 特別研究員制度の助成を受けた．また Shunmiao Xu との有意義な話し合いに感謝する．

参考文献

[1] Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R. and Molina, J.: Controlling data in the cloud: outsourcing computation without outsourcing control, *Proc. ACM Workshop on Cloud Computing Security*, pp. 85–90 (2009).

[2] Vaikuntanathan, V.: Computing blindfolded: New developments in fully homomorphic encryption, *Proc. IEEE 52nd Annu. Symp. Foundations of Computer Science*, pp. 5–16 (2011).

[3] Yi, X., Kaosar, M. G., Paulet, R. and Bertino, E.: Single-database private information retrieval from fully homomorphic encryption, *IEEE Trans. Knowl. Data Eng.*, Vol. 25, No. 5, pp. 1125–1134 (2013).

[4] Boneh, D., Gentry, C., Halevi, S., Wang, F. and Wu, D. J.: Private database queries using somewhat homomorphic encryption, *Proc. Intl. Conf. Applied Cryptography and Network Security*, pp. 102–118 (2013).

[5] Gahi, Y., Guennoun, M. and El-Khatib, K.: A secure database system using homomorphic encryption schemes, *arXiv preprint arXiv:1512.03498* (2015).

[6] Arsovski, I., Chandler, T. and Sheikholeslami, A.: A ternary content-addressable memory (TCAM) based on 4T static storage and including a current-race sensing scheme, *IEEE J. Solid-State Circuits*, Vol. 38, No. 1, pp. 155–158 (2003).

[7] Yu, F. and Katz, R. H.: Efficient multi-match packet classification with TCAM, *Proc. 12th Annu. IEEE Symp. High Performance Interconnects*, pp. 28–34 (2004).

[8] Liu, A. X., Meiners, C. R. and Torng, E.: Packet classification using binary content addressable memory, *Proc. IEEE Conf. Computer Communications*, pp. 628–636 (2014).

[9] Spinney, B. A.: Address lookup in packet data communications link, using hashing and content-addressable memory (1995).

[10] Pagiamtzis, K. and Sheikholeslami, A.: Content-addressable memory (CAM) circuits and architectures: A tutorial and survey, *IEEE J. Solid-State Circuits*, Vol. 41, No. 3, pp. 712–727 (2006).

[11] Khedr, A., Gulak, G. and Vaikuntanathan, V.: SHIELD: Scalable Homomorphic Implementation of Encrypted Data-Classifiers, *IEEE Trans. Comput.*, Vol. 65, No. 9, pp. 2848–2858 (2016).

[12] Ducas, L. and Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second, *Proc. Annu. Intl. Conf. Theory and Applications of Cryptographic Techniques*, pp. 617–640 (2015).

[13] Doröz, Y., Öztürk, E. and Sunar, B.: Accelerating fully homomorphic encryption in hardware, *IEEE Trans. Comput.*, Vol. 64, No. 6, pp. 1509–1521 (2015).

[14] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography, *J. ACM*, Vol. 56, No. 6, p. 34 (2009).

[15] Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem, *Proc. Annu. ACM Symp. Theory of Computing*, pp. 333–342 (2009).

[16] Gentry, C., Sahai, A. and Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based, *Proc. Advances in Cryptology*, pp. 75–92 (2013).

[17] Gentry, C., Peikert, C. and Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions, *Proc. Annu. ACM Symp. Theory of Computing*, pp. 197–206 (2008).

[18] Brakerski, Z. and Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE, *SIAM J. Computing*, Vol. 43, No. 2, pp. 831–871 (2014).

[19] Brakerski, Z., Gentry, C. and Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping, *Proc. Innovations in Theoretical Computer Science Conf.*, pp. 309–325 (2012).

[20] Gentry, C., Goldman, K. A., Halevi, S., Julta, C., Raykova, M. and Wichs, D.: Optimizing ORAM and using it efficiently for secure computation, *Proc. Intl. Symp. Privacy Enhancing Technologies*, pp. 1–18 (2013).

[21] Lindner, R. and Peikert, C.: Better key sizes (and attacks) for LWE-based encryption, *Proc. Cryptographers Track at the RSA Conf.*, pp. 319–339 (2011).