秘密計算上での高速畳み込み及び音声フィルタへの応用

須藤 弘貴[1]　吉田 光樹[2]

概要：近年，データ分析に対する需要が高まる一方で，プライバシーや機密情報漏洩のリスクも強く意識されるようになってきている．このような背景から，秘密計算による機械学習技術が盛んに研究されている．多くの研究では機械学習アルゴリズムの秘密計算上での実現に着目しているが，実際のデータ分析では分析のためにデータを加工する，前処理と呼ばれる工程が必要となる．そのため，実用化に向けては秘密計算上で前処理を実現することも重要になる．
本研究では，秘密計算上での高速畳み込み及び，音声フィルタへの応用を提案する．音声では画像に比べフィルタサイズが大きいケースが典型的であるため，特に高速化が重要となる．(画像: フィルタサイズ数十〜数百程度; 音声: フィルタサイズ数万) しかし，これまで，高速畳み込みアルゴリズムは秘密計算上では実現されていなかった．また，一般的な高速畳み込みアルゴリズムは実数演算が必要などの理由から，秘密計算上での効率的な実現が難しい．そこで，提案手法では NMNT と呼ばれる手法を用いて秘密計算上での高速畳み込み演算を効率的に実現する．
本稿では定義式をそのまま計算する愚直法と提案手法との性能比較も報告する，提案手法は愚直法に比べ 30 倍以上の高速化を達成したことを確認した．

Secure Fast Convolution via MPC with Application to Audio Filter

HIROKI SUDO[1]　YOSHIDA KOUKI[2]

1. はじめに

近年，データ分析に対する需要が高まる一方で，プライバシーや機密情報漏洩のリスクも強く意識されるようになってきている．このような背景から，秘密計算による機械学習技術が盛んに研究されている [1], [2], [3], [4]．これらの研究では機械学習アルゴリズムの秘密計算上での実現に着目しているが，実際のデータ分析では分析のためにデータを加工する，前処理と呼ばれる工程が必要となる．そのため，実用化に向けては秘密計算上で前処理を実現することも重要になる．

本研究では，音声データの前処理に着目し，秘密計算上での高速畳み込み及び，音声フィルタへの応用を提案する．音声フィルタは音声データ分析の前処理として用いられ，ノイズの除去や，逆に反響やノイズを足したデータを作成する (学習時にデータのバリエーションを増やすことで少ないデータから頑強なモデルを作る) のに利用される．音声フィルタの一種である畳み込みフィルタによって，

- 音の反響を再現/除去
- 特定の周波数成分を強調あるいは除去

することが可能である．

1.1 課題

音声では画像に比べフィルタサイズが大きいケースが典型的であるため，特に高速化が重要となる．(画像: フィルタサイズ数十〜数百程度; 音声: フィルタサイズ数万) しかし，これまで，高速畳み込みアルゴリズムは秘密計算上では実現されていなかった．また，一般的な高速畳み込みアルゴリズムは実数演算が必要などの理由から，秘密計算上での効率的な実現が難しい．そこで，提案手法では NMNT と呼ばれる手法を用いて秘密計算上での高速畳み込み演算を効率的に実現する．

[1] NTT 社会情報研究所
[2] 東京大学 情報理工学系研究科

1.2 貢献

- NMNT[5] による秘匿高速畳み込み演算プロトコルを提案.
- 入力の片方が公開値である場合について性能測定を行い，高速な計算&通信コストの低さを両立．既存手法の 30 倍以上速く．実用的な性能であることを確認.

本稿では定義式をそのまま計算する愚直法と提案手法との性能比較も報告する, 提案手法は入力の片方が公開値である場合入力長を $|x(t)| = 80000$，フィルタ長を $|h(t)| = 16000$ のとき 0.16 sec と実用的な性能であり，愚直法に比べ 30 倍の高速化を達成したことを確認した.

2. 準備

2.1 記法

ベクトルを $\vec{x}$ と記述する．ベクトルの i 番目の要素は $\vec{x}_i$ で表す.

- N: 変換長
- p: 素数
- M_p: $2^p - 1$ で表される素数 (メルセンヌ素数)
- $[\![x]\!]$: x のシェア
- $*$: 畳み込み
- $\otimes$: 要素ごとの積
- $|\cdot|$: ベクトルの長さ

2.2 秘密分散

秘密分散とはデータを複数の値に分けて複数パーティに分散する暗号化手法である．本研究では (k, n) 閾値秘密分散によりデータを暗号化する．(k, n) 閾値秘密分散とは，データを n 個のランダムな値 (シェアと呼ばれる) に分割して，k 個以上のシェアを集めると元のデータを復元でき，k 個未満のシェアからは元データの情報を得られないような性質を持つ秘密分散法である．また，シェアが加法に対して準同型性を持つ，線形秘密分散であることを前提とする．本稿では値 x のシェアを $[\![x]\!]$ のように表す．具体的には Shamir 秘密分散 [6] や複製秘密分散 [7] [8] を用いる．秘密分散方式は演算を行う代数系にバリエーションがあるが，メルセンヌ素数 M_p を法とする体上で演算を行う場合に有効である．メルセンヌ素数とは $2^p - 1, p \in \mathbb{N}$ で表される素数のことである．例えば $2^2 - 1 = 3$ はメルセンヌ素数である．M_p が素数であれば p も素数であるが，p が素数であれば M_p も素数であるとは限らない.

2.3 畳み込み

畳み込み演算は音声や画像のフィルタ処理等で用いられる基本的な演算である．関数 $x(t)$ と関数 $h(t)$ の畳み込み演算 $*$ は次のように定義される.

$$y(t) = x(t) * h(t) = \sum_{\tau}^{t} x(t-\tau)h(\tau) \tag{1}$$

$|x(t)| = n$, $|h(t)| = m$ のとき，$t = 0, \ldots, n+m-1$ について計算を行う．ただし，$t < 0 \vee t > n$ のとき $x(t) = 0$, $t < 0 \vee t > m$ のとき $h(t) = 0$ として扱う.

2.4 NMNT

長さ $N = 2^p$ の整数列 $x(t) \in \mathbb{F}_{M_p}^N$ の NMNT(New Mersenne Number Transform) は次式で表される [5].

$$X(k) = \sum_{t=0}^{N-1} x(t)\beta(tk) \mod M_p,\ k = 0, 1, \ldots, N-1 \tag{2}$$

また，逆変換 NMNT^{-1} は変換と同様に次式で定義される.

$$x(t) = N^{-1} \sum_{k=0}^{N-1} X(k)\beta(tk) \mod M_p,\ n = 0, 1, \ldots, N-1 \tag{3}$$

ただし,

$$\beta(tk) = \beta_1(tk) + \beta_2(tk) \tag{4}$$
$$\beta_1(tk) = \mathsf{Re}(\alpha_1 + j\alpha_2)^{tk} \mod M_p \tag{5}$$
$$\beta_2(tk) = \mathsf{Im}(\alpha_1 + j\alpha_2)^{tk} \mod M_p \tag{6}$$
$$\alpha_1 = \pm 2^q \mod M_p \tag{7}$$
$$\alpha_2 = \pm - 3^q \mod M_p \tag{8}$$
$$q = 2^{p-2} \tag{9}$$

d を 2 の倍数とし，長さが N/d の場合，β_1, β_2 は次のように定義される.

$$\beta_1(tk) = \mathsf{Re}((\alpha_1 + j\alpha_2)^d)^{tk} \mod M_p \tag{10}$$
$$\beta_2(tk) = \mathsf{Im}((\alpha_1 + j\alpha_2)^d)^{tk} \mod M_p \tag{11}$$
$$\tag{12}$$

2.4.1 NMNT による畳み込み

NMNT 及び逆 NMNT を通して， $x(t)$ と $h(t)$ の畳み込み $y(t)$ を下記のように計算することができる [5].

$$Y(k) = \mathsf{NMNT}[x(t) * h(t)] = X(k)\Gamma H(k) \tag{13}$$
$$= X(k) \otimes H_{ev}(k) + X(N-k) \otimes H_{od}(k), \tag{14}$$
$$y(t) = \mathsf{NMNT}^{-1}(Y(k)). \tag{15}$$

ただし，Γ は $X(.) \otimes H_{ev}(.) + X(-.) \otimes H_{od}(.)$，$\otimes$ は要素ごとの積, $H(k) = NMNT(h(t))$ である．また，H_{ev}, H_{od}

は偶数部分と奇数部分を表し，以下のように定義される．

$$H_{ev}(k) = H(k) + H(N-k) \times 2^{p-1} \mod M_P \quad (16)$$

$$H_{od}(k) = H(k) - H(N-k) \times 2^{p-1} \mod M_P \quad (17)$$

2.4.2 NMNT の高速化

式 (2) を捉えなおし，小さい入力に対する NMNT に分解することで共通する計算を省き，高速化することができる．[5] 例えば下式のように書き換える高速化手法が提案されている [5]．この場合，NMNT の計算量は $O(N \log_2(N))$ となる．

$$X(2k) = \sum_{t=0}^{N/2-1} [x(t) + x(t+N/2)]\beta(2tk) \mod M_p,$$
$$k = 0, 1, \ldots, N/2-1$$

$$X(2k+1) = \sum_{t=0}^{N/2-1} \{[x(t) - x(t+N/2)]\beta_1(t) + [x(N/2-t) - x(N-t)]\beta_2(t)\}\beta(2tk) \mod M_p,$$
$$k = 0, 1, \ldots, N/2-1$$

2.5 構成要素

2.5.1 加減算，公開値乗算

加減算及び公開値乗算は次のように記述する: $[\![x]\!] + [\![y]\!]$, $c[\![x]\!]$．シェアが加法に対して準同型性を持つ秘密分散を前提としているため，シェア同士の加減算及び定数倍は通信を要さず計算できる．

乗算

シェア同士の乗算は次のように記述する：$\textsc{Mult}(x, y)$．シェア同士の乗算は計算サーバー同士の通信により実現することができる．対象となるシェアの種類や特性が異なる乗算プロトコルがいくつか提案されており，例えば Shamir 秘密分散の乗算プロトコルで代表的な GRR 乗算 [9] や複製秘密分散の乗算プロトコル [10], [11] などが知られている．上述の [9], [10], [11] などの乗算プロトコルではオンラインフェーズ前の中間的なデータ (本稿では中間シェアと呼ぶ) があり，それも加法に対して線形性を持ち，加減算，定数倍が可能であるという特性がある．中間シェアを $\langle \cdot \rangle$，中間シェアを出力するオフラインフェーズを $\textsc{MultOff}$，その後のオンラインフェーズを $\textsc{MultOn}$ と書くこととする．

3. 提案手法

3.1 秘密分散上での NMNT

秘密分散上での NMNT プロトコルを Algorithm 1 に示す．式 (2) を観察すると，$\beta(nk)$ については入力によらない，秘匿する必要のない値であることがわかる．また，M_p 上秘密分散の場合，演算が M_p 上で行われるため，NMNT での剰余演算を別途行う必要はなく，秘密計算上の演算に伴う剰余演算のみで実現することができる．そのため，式 (2) は加算及び公開値乗算といったローカル演算のみで実現することができる．また， 2.4.2 節に述べた分割による高速化も適用することができ，同様に加算及び公開値乗算といったローカル演算のみで実現することができる．

Algorithm 1 秘密分散上での NMNT

Functionality: $[\![X(k)]\!] \leftarrow \mathsf{NMNT}([\![x(t)]\!])$
Input: $[\![x(t)]\!]$
Output: $[\![X(k)]\!]$, where $X(k) = \mathsf{NMNT}(x(t))$.
Parameter: $x(t)$ の長さ N
1: **for** $0 \le k < N$ **do**
2: $\quad [\![X(k)]\!] = \sum_{t=0}^{N-1} [\![x(t)]\!]\beta(tk)$, $k = 0, 1, \ldots, N-1$
3: Output $[\![X(k)]\!]$

逆変換も同様に Algorithm 2 に示すように実現することができる．

Algorithm 2 秘密分散上での NMNT^{-1}

Functionality: $[\![x(t)]\!] \leftarrow \mathsf{NMNT}([\![X(k)]\!])$
Input: $[\![X(k)]\!]$
Output: $[\![x(t)]\!]$, where $[\![x(t)]\!] = \mathsf{NMNT}^{-1}(X(k))$.
Parameter: $X(k)$ の長さ N
1: **for** $0 \le n < N$ **do**
2: $\quad [\![x(t)]\!] = N^{-1} \sum_{k=0}^{N-1} [\![X(k)]\!]\beta(tk)$, $t = 0, 1, \ldots, N-1$
3: Output $[\![x(t)]\!]$

3.2 片方が公開値である場合の畳み込み

片方が公開値である場合の畳み込みプロトコルを Algorithm 3 に示す．$H(k)$ が公開値の場合，暗号文ベクトルと公開値ベクトルとの要素積になるため畳み込みはローカル演算のみで実現することができる．

Algorithm 3 NMNT を用いた畳み込み演算 (片方が公開値である場合)

Functionality: $[\![x(t) * h(t)]\!] \leftarrow [\![x(t)]\!] * h(t)$
Input: $[\![x(t)]\!], h(t)$
Output: $[\![x(t) * h(t)]\!]$.
Parameter: 変換長 N
1: $[\![X(k)]\!] \leftarrow \mathsf{NMNT}([\![x(t)]\!])$
2: $H(k) \leftarrow \mathsf{NMNT}(h(t))$
3: $H_{ev}(k) \leftarrow (H(k) + H(N-k)) \times 2^{p-1}$
4: $H_{od}(k) \leftarrow (H(k) - H(N-k)) \times 2^{p-1}$
5: **for** $0 \le k < N$ **do**
6: $\quad [\![Y(k)]\!] \leftarrow \big[[\![X(k)]\!] \times \{H(k) + H(N-k)\}$
7: $\quad + [\![X(N-k)]\!] \times \{H(k) - H(N-k)\}\big] \times 2^{p-1}$
8: $[\![y(t)]\!] \leftarrow \mathsf{NMNT}^{-1}([\![Y(k)]\!])$
9: Output $[\![y(t)]\!]$

3.3 両方とも暗号文である場合の畳み込み

両方とも暗号文である場合の畳み込みプロトコルを Algorithm 4 に示す．$H(k)$ が暗号文の場合，要素積において暗号文同士の乗算となるが，これは秘匿乗算プロトコルによって実現することができる．

Algorithm 4 NMNT を用いた畳み込み演算 (両方暗号文である場合)

Functionality: $[\![x(t) * h(t)]\!] \leftarrow [\![x(t)]\!] * [\![h(t)]\!]$
Input: $[\![x(t)]\!], [\![h(t)]\!]$
Output: $[\![x(t) * h(t)]\!]$.
Parameter: 変換長 N
1: $[\![X(k)]\!] \leftarrow \mathsf{NMNT}([\![x(t)]\!])$
2: $[\![H(k)]\!] \leftarrow \mathsf{NMNT}([\![h(t)]\!])$
3: $[\![H_{ev}(k)]\!] \leftarrow ([\![H(k)]\!] + [\![H(N-k)]\!]) \times 2^{p-1}$
4: $[\![H_{od}(k)]\!] \leftarrow ([\![H(k)]\!] - [\![H(N-k)]\!]) \times 2^{p-1}$
5: **for** $0 \leq k < N$ **do**
6: $\quad [\![Y(k)]\!] \leftarrow \big[\textsc{Mult}([\![X(k)]\!], [\![H(k)]\!] + [\![H(N-k)]\!])$
7: $\quad +\textsc{Mult}([\![X(N-k)]\!], [\![H(k)]\!] - [\![H(N-k)]\!])\big] \times 2^{p-1}$
8: $[\![y(t)]\!] \leftarrow \mathsf{NMNT}^{-1}([\![Y(k)]\!])$
9: Output $[\![y(t)]\!]$

3.4 最適化

通信を最適化した畳み込みプロトコルを Algorithm 5 に示す．要素積以外はローカル演算で実現できる線形変換であるため，MultOn 前に Algorithm 56 7 行目のように中間シェアを加算してから MultOff を行うことで乗算に要する通信量を乗算 2 回分から 1 回分に半減させることができる．

Algorithm 5 最適化した畳み込み演算

Functionality: $[\![x(t) * h(t)]\!] \leftarrow [\![x(t)]\!] * [\![h(t)]\!]$
Input: $[\![x(t)]\!], [\![h(t)]\!]$
Output: $[\![x(t) * h(t)]\!]$.
Parameter: 変換長 N
1: $[\![X(k)]\!] \leftarrow \mathsf{NMNT}([\![x(t)]\!])$
2: $[\![H(k)]\!] \leftarrow \mathsf{NMNT}([\![h(t)]\!])$
3: $[\![H_{ev}(k)]\!] \leftarrow ([\![H(k)]\!] + [\![H(N-k)]\!]) \times 2^{p-1}$
4: $[\![H_{od}(k)]\!] \leftarrow ([\![H(k)]\!] - [\![H(N-k)]\!]) \times 2^{p-1}$
5: **for** $0 \leq k < N$ **do**
6: $\quad \langle Y(k) \rangle \leftarrow \big[\textsc{MultOff}([\![X(k)]\!], [\![H(k)]\!] + [\![H(N-k)]\!])$
7: $\quad +\textsc{MultOff}([\![X(N-k)]\!], [\![H(k)]\!] - [\![H(N-k)]\!])\big] \times 2^{p-1}$
8: $[\![Y(k)]\!] \leftarrow \textsc{MultOff}\langle Y(k) \rangle$
9: $[\![y(t)]\!] \leftarrow \mathsf{NMNT}^{-1}([\![Y(k)]\!])$
10: Output $[\![y(t)]\!]$

3.5 計算量

3.5.1 NMNT の計算量

式の分割による高速化を行った場合，NMNT にかかるローカル計算量は $O((n+m)\log_2(n+m))$ となる．また，通信は不要であるため，通信量・通信ラウンドは 0 である．

3.5.2 畳み込みの計算量

NMNT 変換後の畳み込みについては，乗算が主要となる．$h(t)$ が公開値の場合，ローカル計算量 $O(n+m)$，通信量・通信ラウンドは 0 である．$h(t)$ が暗号文の場合，ローカル計算量 $O(n+m)$，通信量 $O(n+m)$・通信ラウンドは 1 である．オンライン乗算を $O(n+m)$ 回行うが，それぞれ独立であるため，ローカルでの計算を行った後，通信をまとめて行うことで通信ラウンドを 1 にすることができる．

3.5.3 愚直法との比較

まず，愚直法による畳み込みの計算量を示す．愚直法では式 (1) に従って計算を行い，Algorithm 6 のように計算する．

Algorithm 6 愚直法による畳み込み (片方公開値)

Functionality: $[\![x(t) * h(t)]\!] \leftarrow [\![x(t)]\!] * h(t)$
Input: $[\![x(t)]\!], h(t)$
Output: $[\![y(t)]\!] = [\![x(t) * h(t)]\!]$.
1: **for** $0 \leq t < n+m-1$ **do**
2: $\quad [\![y(t)]\!] \leftarrow [\![0]\!]$
3: **for** $0 \leq t_1 < n$ **do**
4: $\quad$ **for** $0 \leq t_2 < m$ **do**
5: $\quad\quad [\![y(t_1+t_2)]\!] \leftarrow [\![y(t_1+t_2)]\!] + [\![x(t_1)]\!] h(t_2)$
6: Output $[\![y(t)]\!]$

$x(t), h(t)$ どちらかが 0 になる乗算は省略できること，積和はまとめて行うことができることを考慮すると，愚直法による畳み込みの計算量は下記の通りになる: $h(t)$ が公開値の場合，ローカル計算量 $O(nm)$・通信量 $O(n+m)$・通信ラウンド 1 $h(t)$ が暗号文の場合，ローカル計算量 $O(nm)$・通信量 $O(n+m)$・通信ラウンド 1．提案手法と愚直法の計算量を表 1， 2 にまとめた．愚直法と比較して，提案手法はローカル計算量を $O(nm)$ から $O((n+m)log_2(n+m))$ に改善することができている．また，通信量・通信ラウンドは同等である．

表 1: 入力が片方公開値である場合の愚直法と提案手法の計算量

	愚直法	提案手法
ローカル計算量	$O(nm)$	$O((n+m)\log_2(n+m))$
通信量	0	0
通信ラウンド	0	0

表 2: 入力が両方暗号文である場合の愚直法と提案手法の計算量

	愚直法	提案手法
ローカル計算量	$O(nm)$	$O((n+m)\log_2(n+m))$
通信量	$O(n+m)$	$O(n+m)$
通信ラウンド	1	1

3.6 安全性

本稿では 3 台の計算サーバーがあり，そのうちの一台までが semi-honest な攻撃者により侵害されることを想定する．semi-honest な攻撃者はプロトコルには従うが，シェアなどの取得できる情報から秘密情報を盗み見ようとする攻撃者である．下位プロトコルである加減算・乗算プロトコルの秘匿性から，提案手法はこの条件において秘匿性を持つ．

4. 実験

入力の片方が公開値である場合の NMNT による畳み込みプロトコルについて性能評価実験を行った．この場合ローカル計算のみであるため，1 台での計算時間の計測を行った．

4.1 パラメータ

パラメータに関しては以下のとおりである．

- $p = 61$
- $M_p = 2^p - 1$
- $N = 2^{17}$

4.2 実験環境

実験には下記のマシン環境を使用した．

- CPU: Intel(R) Core(TM) i7-6900K CPU @ 3.20GHz
- Memory: 32GB

4.3 実験結果

入力長を $|x(t)| = 80000$，フィルタ長を $|h(t)| = 16000$ とした．サンプリング回数 16000/ sec とすると，それぞれ 5 sec，1 sec にあたる．

この時，愚直法では 4.8 sec を要したのに対し，提案手法では 0.16 sec と約 30 倍高速であった．

5. まとめ

本稿では NMNT[5] による秘匿高速畳み込み演算プロトコルを提案した．漸近的な計算量は愚直法と比較して，ローカル計算量を $O(nm)$ から $O((n+m)log_2(n+m))$ に改善することができた．また，通信量・通信ラウンドは同等である．

さらに，入力の片方が公開値である場合について性能測定を行い，既存手法の 30 倍以上速く．実用的な性能であることを確認した．本技術は，高速かつ通信コストの低い秘匿音声フィルタ等への応用が期待される．

参考文献

[1] Mohassel, P. and Rindal, P.: ABY3: A mixed protocol framework for machine learning, *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pp. 35–52 (2018).

[2] Wagh, S., Gupta, D. and Chandran, N.: SecureNN: 3-Party Secure Computation for Neural Network Training., *Proc. Priv. Enhancing Technol.*, Vol. 2019, No. 3, pp. 26–49 (2019).

[3] Patra, A. and Suresh, A.: BLAZE: blazing fast privacy-preserving machine learning, *arXiv preprint arXiv:2005.09042* (2020).

[4] Wagh, S., Tople, S., Benhamouda, F., Kushilevitz, E., Mittal, P. and Rabin, T.: F: Honest-Majority Maliciously Secure Framework for Private Deep Learning, *Proceedings on Privacy Enhancing Technologies*, Vol. 2021, No. 1, pp. 188–208 (2021).

[5] Boussakta, S. and Holt, A.: New transform using the Mersenne numbers, *IEE Proceedings-Vision, Image and Signal Processing*, Vol. 142, No. 6, pp. 381–388 (1995).

[6] Shamir, A.: How to share a secret, *Communications of the ACM*, Vol. 22, No. 11, pp. 612–613 (1979).

[7] Ito, M., Saito, A. and Nishizeki, T.: Secret sharing scheme realizing general access structure, *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, Vol. 72, No. 9, pp. 56–64 (1989).

[8] Cramer, R., Damgård, I. and Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation, *Theory of Cryptography Conference*, Springer, pp. 342–362 (2005).

[9] Gennaro, R., Rabin, M. O. and Rabin, T.: Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography, *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, PODC '98, Puerto Vallarta, Mexico, June 28 - July 2, 1998* (Coan, B. A. and Afek, Y., eds.), ACM, pp. 101–111 (online), DOI: 10.1145/277697.277716 (1998).

[10] Chida, K., Hamada, K., Ikarashi, D., Kikuchi, R., Kiribuchi, N. and Pinkas, B.: An Efficient Secure Three-Party Sorting Protocol with an Honest Majority, *IACR Cryptology ePrint Archive*, Vol. 2019, p. 695 (online), available from ⟨https://eprint.iacr.org/2019/695⟩ (2019).

[11] Ikarashi, D., Kikuchi, R., Hamada, K. and Chida, K.: Actively Private and Correct MPC Scheme in t<n/2 from Passively Secure Schemes with Small Overhead, *IACR Cryptol. ePrint Arch.*, Vol. 2014, p. 304 (online), available from ⟨http://eprint.iacr.org/2014/304⟩ (2014).