準同型暗号によるバーチャルセキュアプラットフォームの開発
- 暗号化したまま C プログラムを実行する -

1　背景

本プロジェクトのテーマの背景として最もわかりやすいものはクラウドコンピューティングの普及である。クラウドコンピューティングでは、計算資源を提供するクラウドベンダが保有するクラウドサーバに、利用者であるクライアントが計算処理を移譲する。

ここで盗聴者がクラウドサーバに直接アクセスできるような場合を考える。ここで言う盗聴者は悪意ある第三者も含むが、特にクラウドベンダ自身のことを想定している。現在普及しているコンピュータのハードウェアにおいては、少なくとも CPU のチップ内において、プログラムの実行時にはプログラムとその入力であるデータは平文になっていなければならない。そのため、ロジックアナライザやサイドチャネル攻撃などの手法によってハードウェアを流れる電気信号を観測できた場合、平文になっているプログラムやデータを抜き取られる恐れがある。

クラウドコンピューティングにおいては、クラウドサーバの OS 以下すべての層の安全性についてクラウドベンダが責任を負う。それは同時に、クラウドベンダが望めば、専用の器具を取り付けるなど様々な脆弱性をクラウドサーバに仕込めることを意味する。

このような現状では、クライアントはクラウドサーバに脆弱性が存在しないことを信じてクラウドコンピューティングを利用する他ない。これは取りも直さず、クラウドベンダへの信用がクラウドコンピューティングの基盤となっていることを意味している。

2　目的

このような現状において、クラウドベンダにセキュリティ管理を丸投げすることによるリスクを避けつつ、クラウドコンピューティングの恩恵を受け続けるには、ユーザが行うことができる自衛手段、つまりアプリケーションに組み込めるような、ソフトウェアだけで実現できるセキュリティが必要である。

本プロジェクトではそのようなセキュリティの一つとして、プログラムの秘匿がソフトウェアだけで達成可能であることを示すことを目的としている。より具体的には、プログラムとデータの両方を暗号化したまま計算処理を行えるような仕組みをソフトウェアだけで実現する。そうすることによって、たとえ任意の位置の電気信号を読み取ることができたとしても暗号文しか抜き取られないために、計算量的安全性を保証することができる。

この目的を達成する仕組みが、我々が提案するバーチャルセキュアプラットフォーム（Virtual Secure Platform; VSP）である。VSP の特徴は以下の３つである。

1. データだけでなくプログラムも秘匿したまま計算処理ができること

2. 暗号学のみに基づいて安全性を保証できること
3. 高級言語で書いたプログラムが動作すること

このような特徴を兼ね備えた試みは他に例がなく、VSP が世界で初めて実現した。

3　開発の内容

本プロジェクトでは準同型暗号を用いてプログラムの秘匿を実現した。準同型暗号とは図 1 のような関係を満たす暗号である。例えば 0 と 1 の論理積（AND）を計算することを考えると、「0 の暗号」と「1 の暗号」に対してある一定の演算を行えば「0 の暗号」になる。つまり「暗号上での AND」が存在するということである。

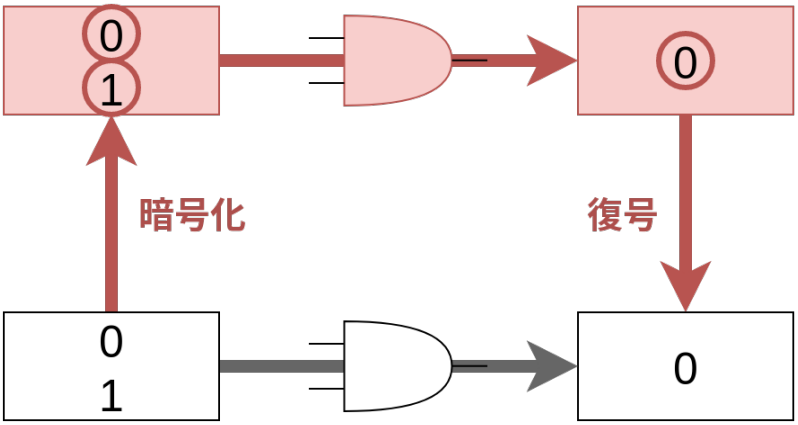

図 1: 準同型暗号

さて平文でのプログラム実行を単純に捉えると、プログラムを CPU に入力し、CPU の動作結果を得る営みであると言える。CPU は論理ゲートと呼ばれる素子が複雑に組み合わさって構成されているが、一つ一つの論理ゲートはとても単純であり、種類も少ない。

そこで本プロジェクトでは、CPU の全ての論理ゲートを、対応する準同型暗号上の演算で置き換えることによって、CPU を準同型暗号上で動作させた。すなわち、暗号化されたプログラムを入力として渡すと、平文で実行したときと同じ結果を暗号化したものを返すような CPU を構成することに成功した。

この際問題になるのは、準同型暗号上での演算は平文上での演算と比べて極めて遅いということである。このことは、今まで本プロジェクトのようなアイデアが実現されてこなかった理由とも推測される。我々はこの問題を準同型暗号の改善のみによってではなく、CPU やその上で動く命令セット（ISA）、及び実行バイナリを生成するコンパイラなど全ての面から改善することで解決しようと試み、一定の成果をあげた。

バーチャルセキュアプラットフォームの概要は図 2 にて示される。ユーザはまず、動かしたい C プログラムをコンパイルして実行バイナリとし、これを秘密鍵を用いて暗号化する。続いて暗号化された実行バイナリを暗号化したまま、秘密鍵を使用

すること無く実行して、暗号化された結果を得る。最後にこれを秘密鍵を用いて復号すると、平文で動作させた時と値が一致する。

このバーチャルセキュアプラットフォームを実現するために、我々は次のような成果物を開発した。

命令セット（ISA）
: 実行バイナリサイズを最小化するために独自に設計した。

C コンパイラ
: 図 2 にて「コンパイル」を行う。LLVM を利用することで、既存の最適化をそのまま適用した。

CPU
: 図 2 にて「実行」に関わる。独自 ISA 用に新たに開発した。論理ゲート数が少なくなるように設計し、準同型暗号ゲートの処理回数を減らした。

準同型暗号ライブラリ
: 図 2 にて「暗号化」・「復号」、及び「実行」に関わる。既存のものにない機能を追加し、アルゴリズムレベルで高速化を図った。

準同型暗号ゲート評価エンジン
: 図 2 にて「実行」を行う。論理ゲート間の依存関係を動的に解決することで、効率的な実行を可能にした。

我々はこの全体の実装に KVSP（Kyoto Virtual Secure Platform）と名付けた。これらの成果は全てオープンソースで公開されている。

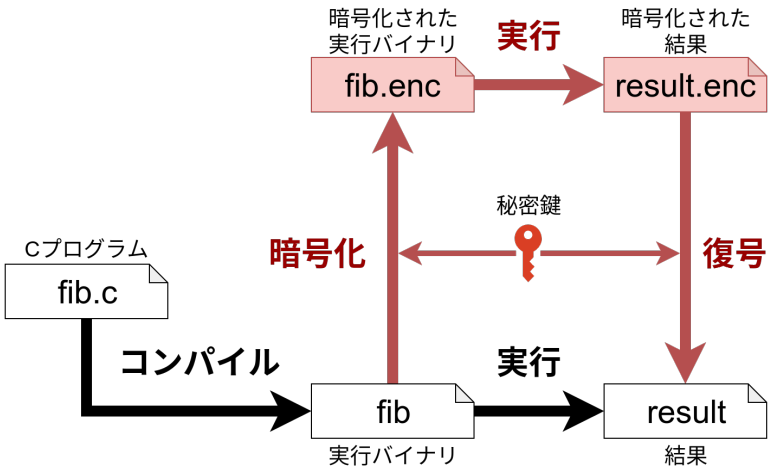

図 2: バーチャルセキュアプラットフォームの概要図

KVSP の現状性能は次のようになっている。NVIDIA Tesla V100 を 1 基搭載したマシンで 1 クロックに約 5.5 秒、8 基搭載したマシンでは約 1.5 秒を要する。また Amazon Web Service の c5.metal インスタンスでは 1 クロックあたり約 2.5 秒で動作する。

4　従来の技術との相違

GarbledCPU・ARM2GC は、Garbled Circuit によって CPU エミュレーションを行うことでプログラムの秘匿を実現するような手法である。前者は MIPS、後者は ARM を ISA として採用している。これらの手法では、実行するクロックが増えれば増えるほどクライアントが生成しなければならない暗号文の量が増え、クライアントの負荷が増大してしまう。他方 VSP は暗号文のサイズが一定、つまりクライアントの負荷が一定であるため、計算処理の移譲により適している。

Intel SGX は Intel 社が提供しているセキュリティ用の機能であり、メモリ上に「Enclave」と呼ばれる暗号的に厳重に保護された領域を生成することで、センシティブデータを保護しつつプログラムを実行する為の CPU の拡張機能である。この機能は Intel 社の提供するハードウェアに依存しており、Intel 社を信用する必要があるという点で VSP とは異なるものとなっている。

5　期待される効果

本プロジェクトで開発した KVSP は、C 言語で書かれたプログラムを暗号化によって秘匿したままクラウドなどで実行するための具体的な手順を与えるものである。プログラムないしデータ、またはその両方を隠したまま計算を実行する秘密計算は、IBM や Microsoft においても研究されている分野である。本プロジェクトの成果は秘密計算の分野において一つの方向性を示すものであり、研究の活性化が期待される。

6　普及の見通し

我々が作った VSP は黎明期のコンピュータの性能に近いものであるが、黎明期のコンピュータ同様に性能向上の余地が大きいものでもある。例えば CPU のマルチコア化のような演算器の集積の恩恵も受けうる。VSP の特徴である「プログラムの秘匿」が無二の価値であって人々を惹きつける限りにおいて、VSP の性能向上の試みは続くだろう。

黎明期のコンピュータが出来上がってから今日まで数十年の歳月が経過している。同様に我々は、VSP は数十年後に普及しうると考えている。

7　クリエータ名（所属）

- 松岡 航太郎（京都大学工学部電気電子工学科）
- 伴野 良太郎（京都大学工学部情報学科）
- 松本 直樹（京都大学工学部情報学科）

（参考）関連 URL

- KVSP：`https://github.com/virtualsecureplatform/kvsp`