機密実行環境で動作するソフトウェアの Zig 言語による開発にむけて

島田 慎太郎[1]　新城 靖[1]

1. はじめに

近年, 機密データを外部から隔離しつつ計算を行う機密実行環境 (Trusted Execution Environment for Confidential Computing) が注目されている．Intel 社の Software Guard Extensions (SGX) は機密実行環境を構築する技術の一種で，CPU に組み込まれたハードウェア機能を用いてエンクレーブと呼ばれる機密実行環境が構築できる．SGX は Trusted Computing Base (TCB) が小さいプログラムを開発可能という利点をもつが，エンクレーブで動作するプログラムの開発は，OS の機能を利用できないため容易ではない．現在広く利用されている開発環境は，C/C++言語および Rust 言語のものに限定される．

本研究では，SGX で動作するソフトウェア開発のすそ野を広げることを目的として，それを Zig 言語 [1] により開発する手法を提案する．Zig は OS がない環境でも動作させやすく，高い型安全性をもち，Zig は強力なメタプログラミングをサポートしている．例えば，関数の引数に `comptime` 引数を用いることでジェネリックな型や関数を実現できる．加えて，C 言語の Application Binary Interface (ABI) との相性がよく，C 言語で記述された既存のライブラリを容易に呼び出せるという利点がある．この論文ではこれらの機能を利用する開発手法を提案する．

2. 一般的な Intel SGX SDK による開発手法

Intel SGX SDK を用いた C/C++による開発では，通常環境とエンクレーブで動作する 2 つのプログラムを独立してビルドする．それらは遠隔手続き呼び出し (RPC) で協調する．通常環境からエンクレーブで動作するプログラムの関数を呼び出すことを ECALL，逆方向に呼び出すことを OCALL という．RPC のインターフェースは Enclave Definition Language (EDL) と呼ばれる言語で記述する．エンクレーブで動作するプログラムには，改ざんを検出するために，ディジタル署名が付加される．

提案手法では，通常環境とエンクレーブで動作するプログラムを両方とも Zig 言語で記述可能にし，Intel SGX SDK と同様にそれらを RPC で協調動作可能にする．

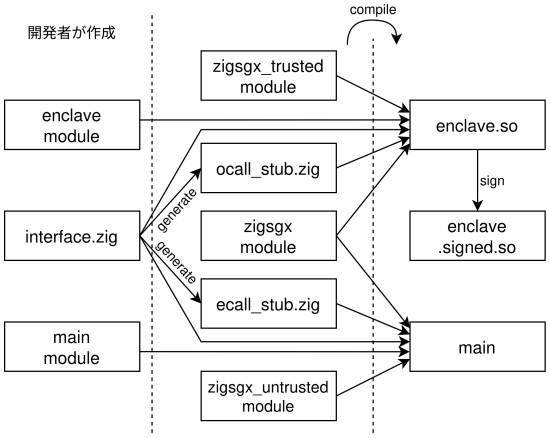


図 1　提案手法の構成

3. 提案手法

Zig では C 言語で書かれた宣言を参照し，C 言語向けのライブラリを使用することができる．しかし，Intel SGX SDK を単純に Zig から利用するだけでは不便な点がある．それは，ECALL/OCALL で呼び出される関数が C 言語から呼び出せるものに限定されるという点である．

本研究では，インターフェースを Zig 言語で記述可能にし，EDL の記述を不要にする．そして，Zig 言語がもつメタプログラミング機能を使用してスタブを生成する．

提案手法の構成を図 1 に示す．開発者は通常環境で動作する**メインモジュール**，エンクレーブで動作する**エンクレーブモジュール**および **interface.zig** を記述する．`interface.zig` には ECALL/OCALL 関数のインターフェースを Zig 言語で記述する．これをスタブ生成器に入力することでクライアント側スタブである `ecall_stub.zig` と `ocall_stub.zig` を生成する．メインモジュールから `ecall_stub.zig` に定義された関数を呼び出すことで ECALL を，エンクレーブモジュールから `ocall_stub.zig` に定義された関数を呼び出すことで OCALL を実行できる．以降，ECALL と OCALL の実現方法は対称的であるため，OCALL に対する説明を省略する．

開発者が ECALL 関数を定義し，呼び出す例を図 2 に示す．ここでは，整数の割り算を行う関数 `div` を ECALL

[1] 筑波大学
University of Tsukuba

関数として定義している．2 行目から 4 行目は構造体のフィールドとしてインターフェースを定義している．戻り値の型である Error!u32 は，基本的には u32 の値が返されるが，代わりにエラーが返される可能性があることを意味する．

7 行目から 15 行目はスタブを介して ECALL を行うメインモジュールの例である．Enclave はスタブが生成する構造体である．この構造体には，自身を受け取る ECALL 関数のスタブが定義され，これをメソッド呼び出しの形で呼び出すことができる．12 行目では，ゼロ除算により 2 行目で定義したエラー値が返却され，catch ブロック内でエラーハンドリングされる．

エンクレーブモジュールのコードを 17 行目から 25 行目に示す．構造体の中で ECALL として呼び出される関数を実装している．本研究では，Zig 言語のコンパイル時実行の機能を用いて ECALL のサーバ側スタブを生成する．図 2 の 23 行目から 25 行目に示すように，エンクレーブモジュールで comptime というキーワードをつけたブロック内で関数 registerEcallFunctions を呼び出す．この関数は，引数に渡されたインターフェースが定義された構造体と実装が記述された構造体の整合性をチェックし，各関数に対して一意の関数 ID を割り当てたテーブルを作成する．サーバ側スタブはそれを参照し，引数で渡された関数 ID に対応する関数を呼び出す．

本研究では，Zig 言語のライブラリ std.zig に含まれる構文解析器 Ast.parse を用いてクライアント側スタブを生成する．スタブ生成器にインターフェースが定義された構造体を与え，抽象構文木を構築する．次に，その構造体の全フィールドに対して名前および型を取得する．そして，関数型をもつフィールド全てに対して，それと同じ型かつ同じ名前のスタブ関数を Enclave 構造体のメソッドとして生成する．最後に，もとの構造体を Enclave 構造体に置き換えたものを ecall_stub.zig に出力する．

クライアント側スタブは対応する関数 ID を引数として，エンクレーブ上のサーバ側スタブを呼び出す．この呼び出しには Intel SGX SDK による ECALL を利用する．この関数のインターフェースは固定であるため，Zig 言語による開発者は EDL を記述する必要はない．

4. 関連研究

HasTEE[2] は Glasgow Haskell Compiler (GHC) を用いてエンクレーブで Haskell プログラムを動作させることを可能としている．HasTEE では，エンクレーブ上でのみアクセスできるセキュアな値とアクセスに制限のない値を型システムで区別する．本研究は，セキュアな値とそうでない値を区別しない点で異なる．

Enarx[3] は WebAssembly プログラムを SGX エンクレーブや AMD SEV などの機密実行環境で動作させることを可能としている．Zig 言語で記述されたプログラムを WebAssembly に変換することで Enarx で実行することができる．Enarx はプログラム全体を機密実行環境で動作させるが，本研究は通常環境で動作する部分とエンクレーブで動作する部分を開発者が選択するという点で異なる．

```
1  // in interface.zig
2  pub const Error = error { ZeroDivision };
3  pub const EcallInterface = struct {
4      div: fn (a: u32, b: u32) Error!u32,
5  };
6  // in main module
7  pub fn main() !void {
8      const enclave =
9          try Enclave.init("enclave.signed.so");
10     defer enclave.deinit();
11     _ = try enclave.div(10, 3); // returns 3
12     _ = enclave.div(10, 0) catch |err| {
13         // error handling
14     };
15 }
16 // in enclave module
17 pub const ecall_impl = struct {
18     pub fn div(a: u32, b: u32) Error!u32 {
19         if (b == 0) return Error.ZeroDivision;
20         return a / b;
21     }
22 };
23 comptime {
24     registerEcallFunctions(EcallInterface,
           ecall_impl);
25 }
```

図 2 ECALL 関数を定義し，呼び出す例

5. おわりに

この論文では，Zig 言語を用いた SGX エンクレーブで動作するソフトウェアの開発手法を提案した．提案手法では，環境間の RPC のインターフェースを Zig 言語で記述し，Zig 独自の型を扱う関数を ECALL/OCALL として呼び出すことができる．今後の課題はリモートアテステーションと TLS による安全な通信路の確立である．

謝辞 本研究は部分的に JSPS 科研費 21K19756 と 25K03104 の助成を受けた．

参考文献

[1] Zig Software Foundations: Zig Programming Language. 入手先 ⟨https://ziglang.org/⟩ （参照 2026-10-22）
[2] Abhiroop Sarkar, Robert Krook, Alejandro Russo, and Koen Claessen.: *HasTEE: Programming Trusted Execution Environments with Haskell.*, Proceedings of the 16th ACM SIGPLAN International Haskell Symposium, pages 72-88, 2023.
[3] The Enarx Project: Enarx, Confidential Computing with WebAssembly. 入手先 ⟨https://enarx.dev/docs/technical/introduction⟩ （参照 2026-10-22）