

Corda Tech Meetup 秋の陣

SBI^{r3}.
Japan

—● **Intel[®] SGX**

—● **Conclave**

インテルSGXのアプリケーション開発キット 「Conclave」のご紹介

SBI R3 Japan株式会社

2021年11月25日

秘密計算を実現する **conclave**

サービスプロバイダーやクラウドベンダーからデータを保護したまま処理を実行

- ✓ Conclaveは、秘密計算を行うためのハードウェアであるIntel SGXに、アプリケーションの安全な実行環境(TEE)の実装を容易にするSDK(Software Development Kit)
- ✓ ホストアプリケーションの内部にEnclaveという隔離領域を作り、処理を実行
- ✓ 秘匿データにアクセスできるのはこの内部プログラムだけなので、サービスプロバイダーやクラウドベンダーなどからは見えない。またプログラム自体の改ざん不可

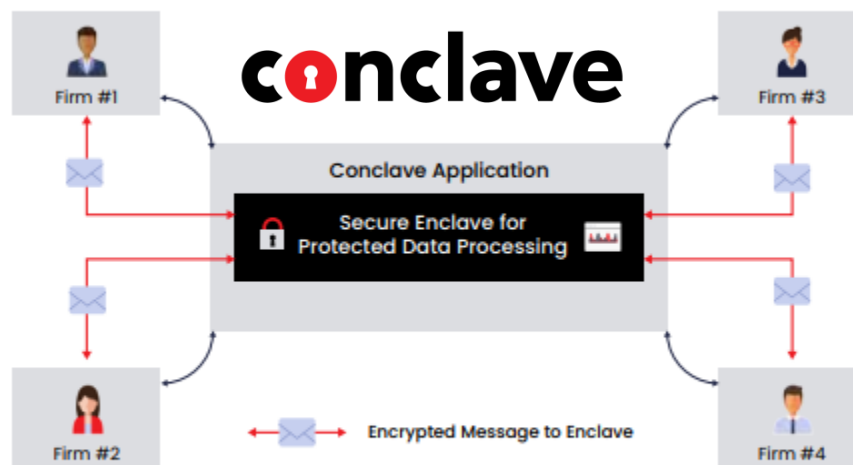


Figure 1: Multi-party data pooling

Conclave公式サイト : [リンク](#)

秘密計算により内容の結果だけを取得：

入力された
原材料情報

原材料	
小麦粉.....	10g
塩.....	5g
そば粉.....	20g
ねぎ.....	5g
鰹節.....	1g

レシピ詳細を見せたくない

情報の格納は暗号化

原材料
dy374899095tsf35s56ac
sds3sjudte5h998780987
hjs63473902yshdtge633
jfh56352938654378thw
2ujd67wp9hvcjdjsker667

アレルギー情報

小麦粉は入っていますか？

小麦粉は入っています

プラットフォーム上の
アプリケーション

あらかじめ合意した質問に対して
(計算して) 回答だけを表示
中身のデータを見ることができない

栄養成分表示

エネルギー	360 kcal
たんぱく質	11.2 g
脂質	3.0 g
炭水化物	75.0 g
ナトリウム	14 mg
ビタミンB1	0.45 mg
...	
(重量100g当たり)	

原材料は分からないが
栄養成分表示に変換

Conclaveのコンポーネント

Conclaveアーキテクチャーを構成する3つのコンポーネント

Enclave

Enclave(エンクレーブ : 飛び地) とはインテル® SGXマシンの保護されたメモリ空間で動作し、外部から隔離されたプログラムのことです。プライベートなデータを秘密裏に処理し、処理結果を返します。ホストと同じJVMを共有しない専用のサブJVM (GraalVM) にロードされます。

Host

エンクレーブは隔離されており、クライアントはエンクレーブに直接アクセスすることはできず、メッセージをエンクレーブに届けるのはホストになります。ホストマシンでは以下のようなホストプログラムを実行します。

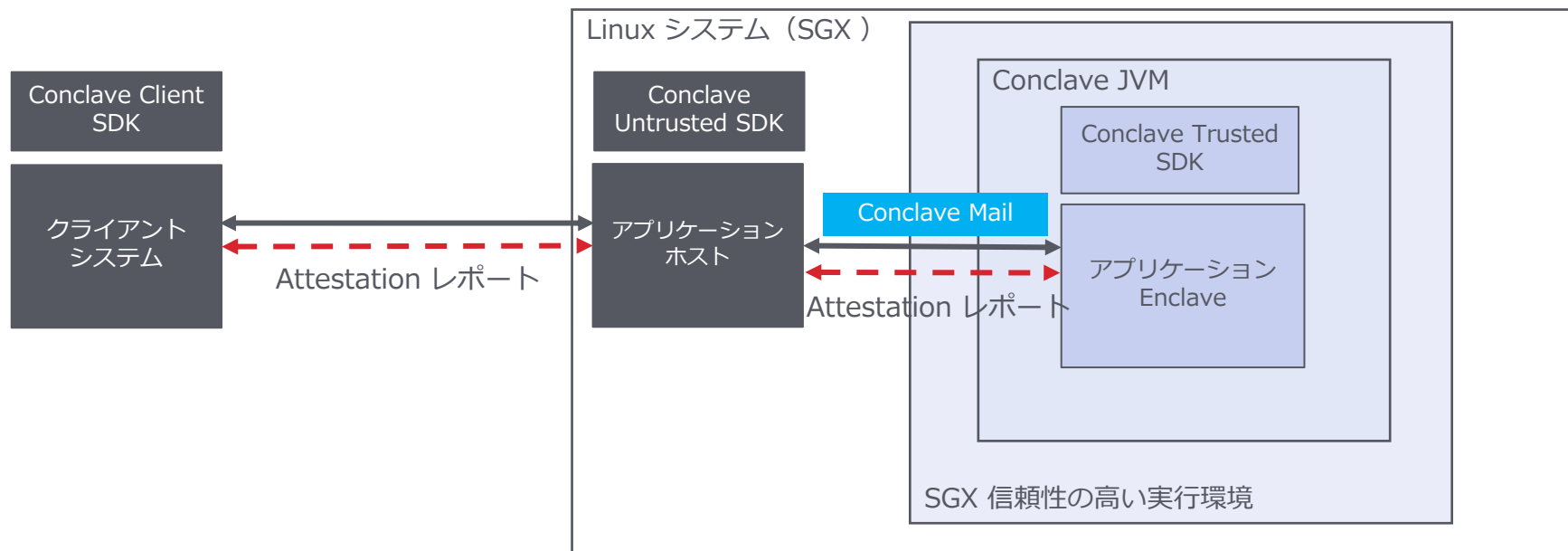
- エンクレーブのプログラムを読み込む
- エンクレーブに必要なリソースの提供
- クライアントとの間の代理人としての役割

Client

クライアントは、機密データをエンクレーブに送信し、ホスト経由で処理するプログラムです。Conclaveには、エンクレーブとクライアント間のコミュニケーションを容易にするためのMail APIが付属しています。

アーキテクチャ イメージ図

Conclaveにより信頼性の高い実行環境を容易に構築可能



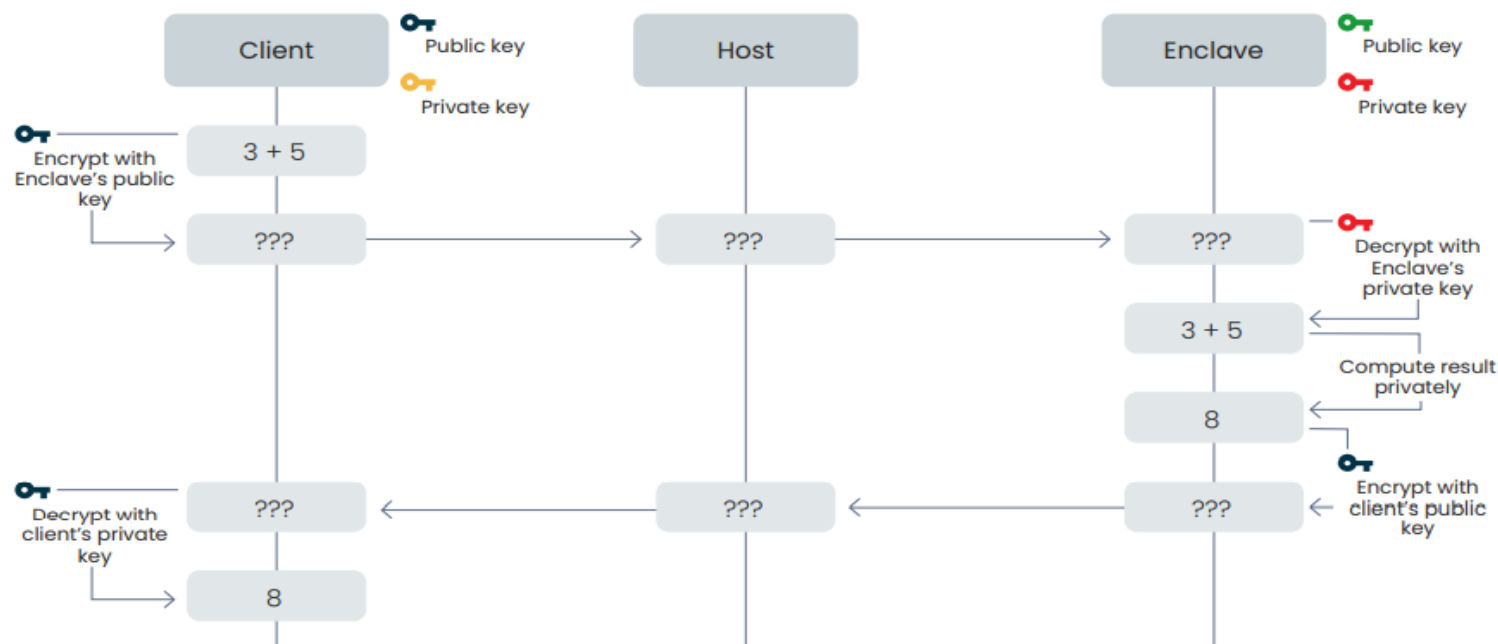
- Conclaveは専門家でなくても使用可能
 - ✓ Intel SGXの専門知識不要
 - ✓ SDKクラスでTEE技術をシンプルに
 - ✓ 明確なドキュメントやチュートリアルが用意されている

■ 信頼されていない
■ 信頼できる
■ 暗号化

暗号化メッセージング

Conclaveの各コンポーネントは信頼性の高い暗号で通信します

- エンクレイブとクライアントは、キーペアを使用して、両者間で交換される情報を暗号化および復号化します。
- クライアントは、エンクレイブに送信された情報をエンクレイブの公開鍵を使って暗号化し、エンクレイブの秘密鍵を使ってのみ復号できるようにします。
- このため、情報を中継する役割を担うホストは、その情報を読み取ったり改ざんしたりすることができません。
- 同様に、エンクレイブはクライアントの公開鍵を使用して処理結果を暗号化するため、クライアントのみが復号化して読み取ることができます。



Conclaveの強み

Conclaveは開発者フレンドリーな秘密計算SDKです

秘密計算のハードルの高さ

- Intel SGXにTEEを実装するためには、低レベル言語や暗号技術に対する専門知識が必要
- 専門人材の確保が必要、または開発者への学習コスト高

Conclaveにより秘密計算を使ったサービス構築が容易に

1. Conclaveは専門家でなくても使用可能
 - Intel SGXの専門知識不要
 - SDKクラスでTEE技術をシンプルに
 - 明確なドキュメントやチュートリアルが用意されている
2. GraalVMとの統合
 - Java、Kotlin、Scalaなどの標準的なJVM言語や、Javascriptなどのスクリプト言語でアプリを開発可能
3. Windows、Linux、MacOS対応

リモート認証(Remote Attestation)

エンクレイブの信憑性を検証するために、「リモート認証」が使用されます

どんなプログラムもランダムなキーペアを使用して正当なエンクレイブを騙ることができます。クライアントにとって重要なことは、エンクレイブであると宣伝しているソフトウェアが実際に正当なエンクレイブであることを確認することが必要です

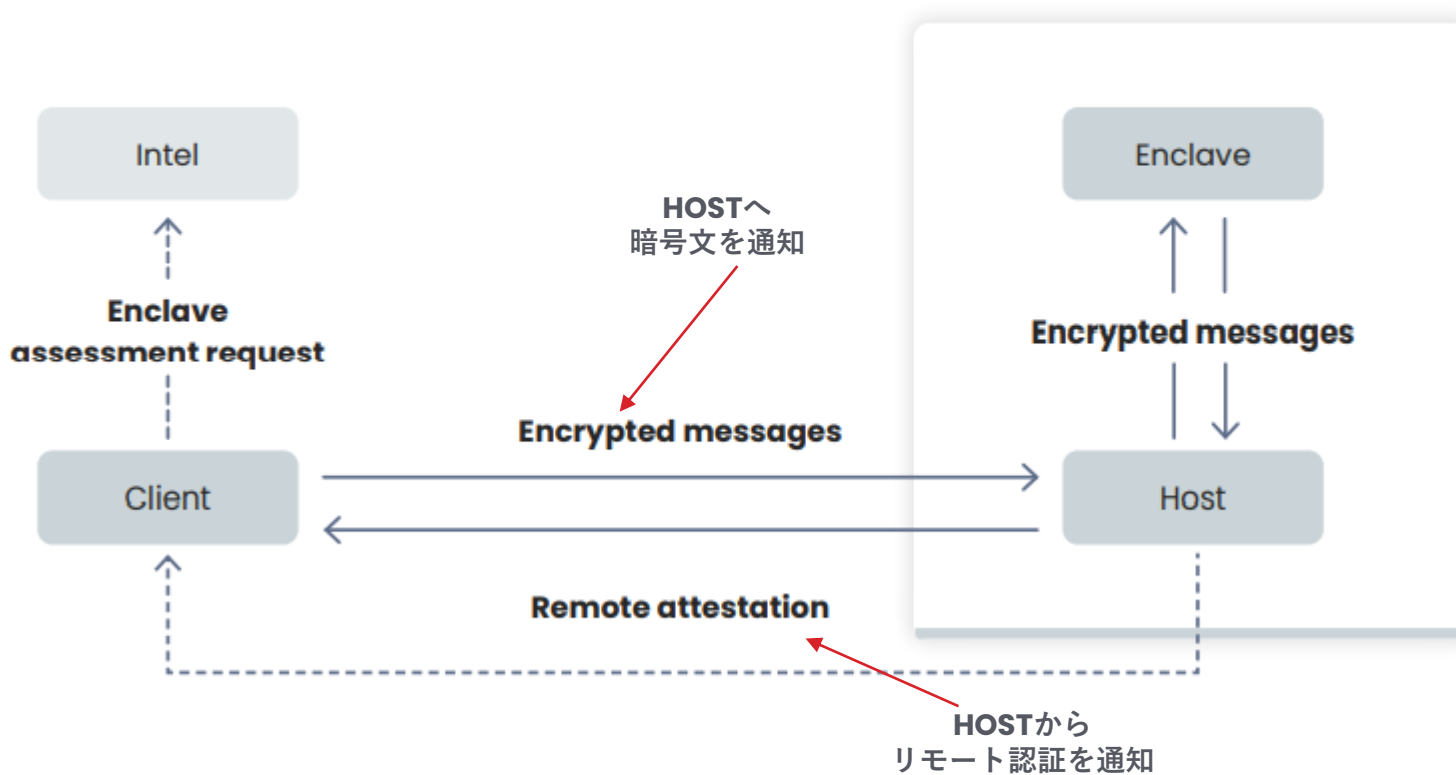
リモート認証とはエンクレイブを確認するための重要な情報を含んだデータのことです。測定値(measurement)が含まれています。測定値は、特別なツールを使用して生成された一意のハッシュで、ハッシュはenclaveにロードされたfat-jarから生成されます。測定値はenclaveのソースコードをコンパイルすることで検証できます。(同じソースコードの複数のビルドが同じ測定値を生成する)

リモート認証に加えて、クライアントはインテル® SGX にエンクレイブの評価を依頼し、エンクレイブが安全であるかどうかを確認することができます。インテル® SGX は、エンクレイブのハードウェアが最新のバージョンを使用しているか、セキュリティ上の脆弱性があるかなど、重要な情報を提供します。

Conclaveのアーキテクチャー図

Conclaveの3つのコンポーネントの関係は下記の概念図になります。

Conclave application architecture



Conclaveメール

開発者がクライアントとエンクレイブ間の通信を容易にするMail API

Encryption

クライアントとエンクレイブ間のメッセージをAES暗号を用いてエンドツーエンドで暗号化。

Headers

メールには、暗号化されていない改ざん防止されたヘッダーがあり、これを使ってメッセージを結びつけることができます。また、トピックとシーケンス番号を使用することで、メッセージの順番を変えたり、メッセージを取りこぼしを防ぎます。

Authentication

メールは、メッセージが特定の鍵の所有者から送られてきたことを証明するとともに、送信先の公開鍵に暗号化することができます。

Framing

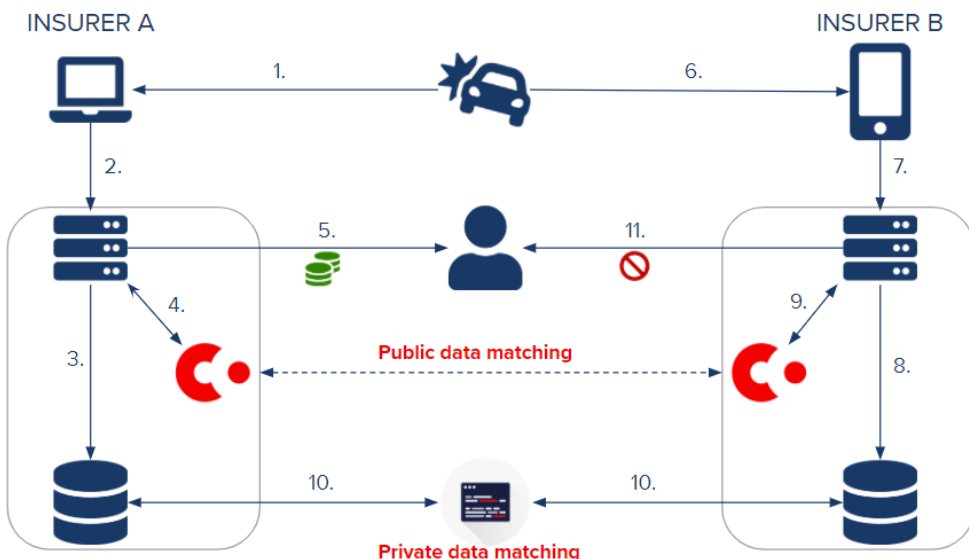
メールは、メッセージの始まりと終わりが簡単に識別できるように区切りますが、その上にフレーミングを実装する必要はありません。

Conclave事例①ClaimShare（保険金の二重支払防止）

概要

Insurance EuropeとKPMGの調査によると、2017年だけでもヨーロッパでの不正請求の推定総額は130億ユーロでした。各保険会社はデータ分析によって防止に努めているものの、個人情報の共有に関する規制が理由で、自社データの分析に限られていました。

しかし、Cordaのプライバシー設計により、企業横断的な新しい不正検知のモデルが実現しました。



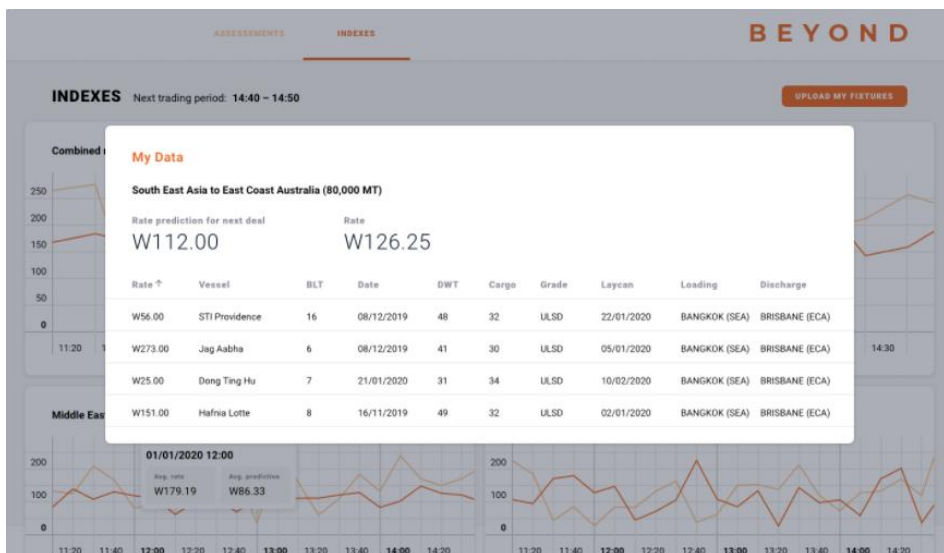
サービス名	ClaimShare
開発パートナー	
ユースケース	損害保険／生命保険
サービス特徴	<ul style="list-style-type: none"> 保険会社は保険金の支払い履歴をCordaに記録します。それにより、新たに保険金請求を受けた保険会社は、別の保険会社で既に支払いが行われているか検証できます。疑わしい請求が検知された場合は、より詳細な比較検証を行います。詳細情報も一致する場合は不正請求と見なして保険金は支払われません。 R3社のConclave(Intel SGXを利用してConfidential Computingの機能を実装したもの)を使用することにより、データを他の保険会社に公開することなく検証できます。
紹介動画	Link
Next Step	<ul style="list-style-type: none"> R3社とB3i社がスポンサーを務めるCorda Challenge 2020 保険部門で優勝 自動車以外の保険にも適用予定

Conclave事例②Beyond（機密取引プラットフォーム）

課題

トレーダーはマーケットで流動性を求めるが、自分の取引条件や価格条件がマーケットに悪影響を及ぼす可能性があるため、それらを公開したくない場合がある。

また、取引所の運営者とその従業員は、取引相手の身元や取引条件など、悪用される可能性のある重要な商業情報にアクセスすることができる。




サービス名	Beyond
開発パートナー	 appliedblockchain
ユースケース	キャピタルマーケット
サービス特徴	<ul style="list-style-type: none"> Conclaveにより機密取引プラットフォームを実現 取引履歴、取引量、価格データは、取引所運営者から秘匿された状態で取引が処理される マッチングやプライシングのアルゴリズムの透明性の担保 取引所運営者に個別の取引データを開示することなく、コンプライアンス管理を実行 トレーダーは取引ユーザインタフェースやAPIを介して取引可能
Webサイト	Link
Next Step	<ul style="list-style-type: none"> コモディティ市場でテスト中

秘密計算の活用が見込まれる分野

- AML (Anti Money Laundering)
- ダークプール
- プライベートオークション
- AIや機械学習を使ったプライベートデータ分析
- 会社間でプライベートデータを収集して分析 など

秘密計算技術比較

種類		概要	課題
ソフトウェアによる秘密計算	ゼロ知識証明 (ZKP)	<ul style="list-style-type: none"> ✓ データを公開することなく、データへのアクセス権を持つ事を証明する 	<ul style="list-style-type: none"> • 計算能力への高い要求 • スケーラビリティ • 実装が難しい (任意のアルゴリズムを実装できるわけではない)
	準同型暗号化 (HE/FHE)	<ul style="list-style-type: none"> ✓ 部分準同型暗号(HE) ✓ 完全準同型暗号(FHE) 	<ul style="list-style-type: none"> • 計算コストが高い (クラウド移行すら厳しい) • 完全準同型暗号は、平文操作の除外が難しく、ユースケースに制限
	セキュア・マルチ・パーティ・コンピュテーション (sMPC)	<ul style="list-style-type: none"> ✓ 各当事者がデータの断片だけを保持する「秘密の共有」に依存 ✓ バラバラのデータの一部が知られても元のデータは秘匿可 	<ul style="list-style-type: none"> • ユースケースごとのカスタムアルゴリズム構築 • 高い通信コスト / 高い計算コスト
ハードウェアによる秘密計算		<ul style="list-style-type: none"> ✓ ハードウェア依拠 ✓ 様々なアルゴリズムに対応 ✓ コード実行とメモリを他から分離する ✓ TEE(Trusted Execution Environment)を構築 	<ul style="list-style-type: none"> • 低レベル言語や暗号技術に対する専門知識が必要 <div style="text-align: right; margin-top: 10px;">  で実装を容易に </div>

ハードウェア (TEE)による秘匿計算比較

① Intel SGX

(ア) Intel SGXはビジネスロジックのセンシティブな部分だけ、つまり最小限の信頼を置くことに重点を置いています。

(イ) マルチ・パーティのコンピューティング機能に対する強力な監査プロセス

(ウ) SGX TCBリカバリー・プロセスが、ソフトウェア・バグや、**サイド・チャンネル・アタック**の対処に成功

② AMD SEV

(ア) AMD SEVはクラウド上のVM保護を強く意識している

(イ) マルチ・パーティのコンピューティング用途には不向き

(ウ) ソフトウェアに致命的なバグや、セキュリティ上の欠陥が発見された場合に、ソフトウェアのリカバリーができない

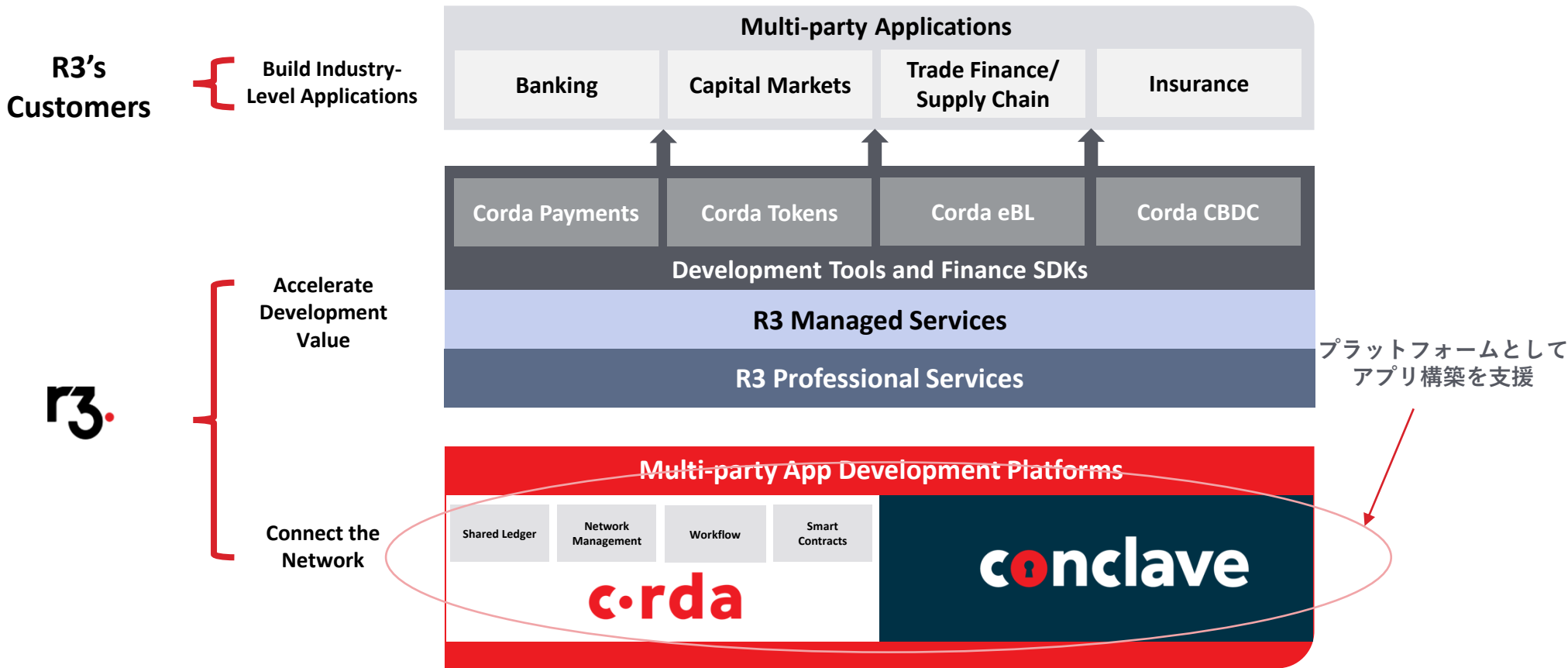
③ ARM Trustzone

(ア) ARM Trustzoneはリモート認証をサポートしていません。

(イ) 主に携帯電話メーカーにおいて、ユーザーから携帯電話をロックするために使われます。

(ウ) モバイルOSに重点を置いています

Cordaとならばアプリケーション基盤の位置づけ



参考 Windows wsl2での サンプル実行例

Sampleの実行例(Windows WSL2 ubuntu)

予めOpenjdkとbuild-essentialを入れておきます

Server側の立ち上げ

```
hotta@SR19-005: ~/src/conclave-sdk-1.1/hello-world
hotta@SR19-005:~/src/conclave-sdk-1.1/hello-world$ ls
LICENSE.txt  build.gradle  enclave  gradle.properties  gradlew.bat  settings.gradle
README.md   client       gradle  gradlew            host         signing
hotta@SR19-005:~/src/conclave-sdk-1.1/hello-world$ ./gradlew host:run

> Task :enclave:generateEnclaveMetadataSimulation
Enclave code hash:    C8F763460954775CA20DF767EC8F447CA23DB73E4F05BC1235B4628843848B43
Enclave code signer: 4924CA3A9C8241A3C0AA1A24A407AA86401D2B79FA9FF84932DA798A942166D4

> Task :host:run
This platform does not support hardware enclaves: SGX_DISABLED_UNSUPPORTED_CPU: SGX is not supported by the CPU in this
system
Listening on port 9999. Use the client app to send strings for reversal.
<=====--> 95% EXECUTING [47s]
> :host:run
```

Sampleの実行例(Windows WSL2 ubuntu)

Client側の実行

```
hotta@SR19-005: ~/src/conclave-sdk-1.1/hello-world
hotta@SR19-005:~/src/conclave-sdk-1.1$ ls
LICENSE.txt  README.md  UPGRADING.md  cordapp  docs  hello-world  licenses  repo  scripts  sources.zip
hotta@SR19-005:~/src/conclave-sdk-1.1$ cd hello-world/
hotta@SR19-005:~/src/conclave-sdk-1.1/hello-world$ ./gradlew client:run --args="HORITA RIHO"
Starting a Gradle Daemon, 1 busy Daemon could not be reused, use --status for details

> Task :client:run
Attempting to connect to localhost:9999
Connected to Remote attestation for enclave C8F763460954775CA20DF767EC8F447CA23DB73E4F05BC1235B4628843848B43:
- Mode: SIMULATION
- Code signing key hash: 4924CA3A9C8241A3C0AA1A24A407AA86401D2B79FA9FF84932DA798A942166D4
- Public signing key: 302A300506032B657003210007C159388855F2ECD0B34C36C31F00ED276D144F1DC077D294F3F28F542E98B8
- Public encryption key: 4D92642BF5C7B93DDB912D809230DE3BFE09531F9095617BF3E90D720F84E151
- Product ID: 1
- Revocation level: 0

Assessed security level at 2021-11-24T21:55:54.089Z is INSECURE
- Enclave is running in simulation mode.
Sending the encrypted mail to the host.
Reading reply mail of length 182 bytes.
Enclave reversed 'HORITA RIHO' and gave us the answer 'OHIR ATIROH'

BUILD SUCCESSFUL in 6s
2 actionable tasks: 1 executed, 1 up-to-date
hotta@SR19-005:~/src/conclave-sdk-1.1/hello-world$
```

Sampleの実行例(Windows WSL2 ubuntu)

server側(client 実行後)

```
hotta@SR19-005: ~/src/conclave-sdk-1.1/hello-world
hotta@SR19-005:~/src/conclave-sdk-1.1/hello-world$ ls
LICENSE.txt  build.gradle  enclave  gradle.properties  gradlew.bat  settings.gradle
README.md   client       gradle  gradlew           host       signing
hotta@SR19-005:~/src/conclave-sdk-1.1/hello-world$ ./gradlew host:run
Starting a Gradle Daemon (subsequent builds will be faster)

> Task :enclave:generateEnclaveMetadataSimulation
Enclave code hash:   C8F763460954775CA20DF767EC8F447CA23DB73E4F05BC1235B4628843848B43
Enclave code signer: 4924CA3A9C8241A3C0AA1A24A407AA86401D2B79FA9FF84932DA798A942166D4

> Task :host:run
This platform does not support hardware enclaves: SGX_DISABLED_UNSUPPORTED_CPU: SGX is not supported by the CPU in this system
Listening on port 9999. Use the client app to send strings for reversal.
Remote attestation for enclave C8F763460954775CA20DF767EC8F447CA23DB73E4F05BC1235B4628843848B43:
- Mode: SIMULATION
- Code signing key hash: 4924CA3A9C8241A3C0AA1A24A407AA86401D2B79FA9FF84932DA798A942166D4
- Public signing key: 302A300506032B657003210007C159388855F2ECD0B34C36C31F00ED276D144F1DC077D294F3F28F542E98B8
- Public encryption key: 4D92642BF5C7B93DD912D809230DE3BFE09531F9095617BF3E90D720F84E151
- Product ID: 1
- Revocation level: 0

Assessed security level at 2021-11-24T21:55:54.089Z is INSECURE
- Enclave is running in simulation mode.

Reversing Hello World!: !dlroW olleH

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.6.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 2m 13s
19 actionable tasks: 6 executed, 13 up-to-date
hotta@SR19-005:~/src/conclave-sdk-1.1/hello-world$
```

Visual Studio Code interface showing a Java project named 'Client.java' in a WSL environment (Ubuntu-20.04). The Explorer view on the left shows the project structure, including folders like 'client', 'enclave', and 'src'. The main editor displays the code for 'Client.java', which is a public class with a main method. The code imports 'java.nio.charset.StandardCharsets' and 'java.security.PrivateKey'. The main method contains comments and logic for connecting to a host (localhost:9999) and receiving an attestation. The status bar at the bottom indicates the current line is 41, column 23, with 4 spaces and tabs, using UTF-8 encoding.

```
Client.java 1 x
client > src > main > java > com > r3 > conclave > sample > client > Client.java > Client > main(String[])
14 import java.nio.charset.StandardCharsets;
15 import java.security.PrivateKey;
16
17 public class Client {
    Run | Debug
18     public static void main(String[] args) throws Exception {
19         // This is the client that will upload secrets to the enclave for processing.
20         //
21         // In this simple hello world app, we just connect to a TCP socket, take the EnclaveInstanceInfo we're sent
22         // and transmit an encrypted string. The enclave will reverse it and send it back. You can use this sample
23         // as a basis for your own apps.
24
25         if (args.length == 0) {
26             System.err.println("Please pass the string to reverse on the command line using --args=\"String to Reverse\"");
27             return;
28         }
29         String toReverse = String.join(" ", args);
30
31         // Connect to the host, it will send us a remote attestation (EnclaveInstanceInfo).
32         DataInputStream fromHost;
33         DataOutputStream toHost;
34         while (true) {
35             try {
36                 System.out.println("Attempting to connect to localhost:9999");
37                 Socket socket = new Socket();
38                 socket.connect(new InetSocketAddress(InetAddress.getLoopbackAddress(), 9999), 5000);
39                 fromHost = new DataInputStream(socket.getInputStream());
40                 toHost = new DataOutputStream(socket.getOutputStream());
41                 break;
42             } catch (Exception e) {
43                 System.err.println("Retrying: " + e.getMessage());
44                 Thread.sleep(2000);
45             }
46         }
47
48         byte[] attestationBytes = new byte[fromHost.readInt()];
49         fromHost.readFully(attestationBytes);
50         EnclaveInstanceInfo attestation = EnclaveInstanceInfo.deserialize(attestationBytes);
```

Visual Studio Code interface showing a Java project named 'HELLO-WORLD [WSL: UBUNTU-20.04]'. The Explorer view on the left shows the project structure, including folders like 'client', 'enclave', and 'host', and files like 'Client.java', 'ReverseEnclave.class', and 'MockTest.class'. The main editor displays the code for 'Client.java' with line numbers 49 to 86. The code includes comments and logic for connecting to an enclave, checking its identity, and sending encrypted mail. The status bar at the bottom indicates 'WSL: Ubuntu-20.04' and '0 4' errors.

```

49 fromHost.readFully(attestationBytes);
50 EnclaveInstanceInfo attestation = EnclaveInstanceInfo.deserialize(attestationBytes);
51
52 // Check it's the enclave we expect. This will throw InvalidEnclaveException if not valid.
53 System.out.println("Connected to " + attestation);
54 // Three distinct signing key hashes can be accepted.
55 // Release mode:      360585776942A4E8A6BD70743E7C114A81F9E901BF90371D27D55A241C738AD9
56 // Debug/Simulation mode: 4924CA3A9C8241A3C0AA1A24A407AA86401D2B79FA9FF84932DA798A942166D4
57 // Mock mode:          0000000000000000000000000000000000000000000000000000000000000000
58 EnclaveConstraint.parse("S:360585776942A4E8A6BD70743E7C114A81F9E901BF90371D27D55A241C738AD9 "
59     + "S:4924CA3A9C8241A3C0AA1A24A407AA86401D2B79FA9FF84932DA798A942166D4 "
60     + "S:0000000000000000000000000000000000000000000000000000000000000000 PROD:1 SEC:INSECURE").check(attestation);
61
62 // Now we checked the enclave's identity and are satisfied it's the enclave from this project, we can send mail
63 // to it.
64
65 // We will need to provide our own private key whilst encrypting, so the enclave gets our public key and can
66 // encrypt a reply. If you already have a long term key then you can use that and the enclave can then
67 // use the public key as a form of identity.
68 PrivateKey myKey = Curve25519PrivateKey.random();
69
70 // For encrypting mail to the enclave we need to create a PostOffice from the enclave's attestation object.
71 // The post office will manage sequence numbers for us if we send more than one mail, which allows the enclave
72 // to check that no mail has been dropped or reordered by the host.
73 //
74 // We use a topic value of "reverse" but any will do in this example. However, mail related to each other and
75 // which need to be ordered must use their own topic. Topics are scoped to the sender key and so multiple clients
76 // can use the same topic without overlapping with each other.
77 //
78 // In this example it doesn't matter as we only send one mail with a random key, but in general it is very
79 // important to use the same post office instance when encrypting mail with the same topic and private key.
80 PostOffice postOffice = attestation.createPostOffice(myKey, "reverse");
81
82 byte[] encryptedMail = postOffice.encryptMail(toReverse.getBytes(StandardCharsets.UTF_8));
83
84 System.out.println("Sending the encrypted mail to the host.");
85
86 toHost.writeInt(encryptedMail.length);

```

Visual Studio Code interface showing a Java file named `Client.java` in a project structure. The file content is as follows:

```

client > src > main > java > com > r3 > conclave > sample > client > Client.java > Client > main(String[])
66 // encrypt a reply. If you already have a long term key then you can use that and the enclave can then
67 // use the public key as a form of identity.
68 PrivateKey myKey = Curve25519PrivateKey.random();
69
70 // For encrypting mail to the enclave we need to create a PostOffice from the enclave's attestation object.
71 // The post office will manage sequence numbers for us if we send more than one mail, which allows the enclave
72 // to check that no mail has been dropped or reordered by the host.
73 //
74 // We use a topic value of "reverse" but any will do in this example. However, mail related to each other and
75 // which need to be ordered must use their own topic. Topics are scoped to the sender key and so multiple clients
76 // can use the same topic without overlapping with each other.
77 //
78 // In this example it doesn't matter as we only send one mail with a random key, but in general it is very
79 // important to use the same post office instance when encrypting mail with the same topic and private key.
80 PostOffice postOffice = attestation.createPostOffice(myKey, "reverse");
81
82 byte[] encryptedMail = postOffice.encryptMail(toReverse.getBytes(StandardCharsets.UTF_8));
83
84 System.out.println("Sending the encrypted mail to the host.");
85
86 toHost.writeInt(encryptedMail.length);
87 toHost.write(encryptedMail);
88
89 // Enclave will mail us back.
90 byte[] encryptedReply = new byte[fromHost.readInt()];
91 System.out.println("Reading reply mail of length " + encryptedReply.length + " bytes.");
92 fromHost.readFully(encryptedReply);
93 // The same post office will decrypt the response.
94 EnclaveMail reply = postOffice.decryptMail(encryptedReply);
95 System.out.println("Enclave reversed '" + toReverse + "' and gave us the answer '" + new String(reply.getBodyAsBytes()) + "'");
96
97 toHost.close();
98 fromHost.close();
99 }
100 }
101

```

The status bar at the bottom indicates: WSL: Ubuntu-20.04, 0 errors, 4 warnings, and the current cursor position is line 41, column 23.

SBI R3.
Japan