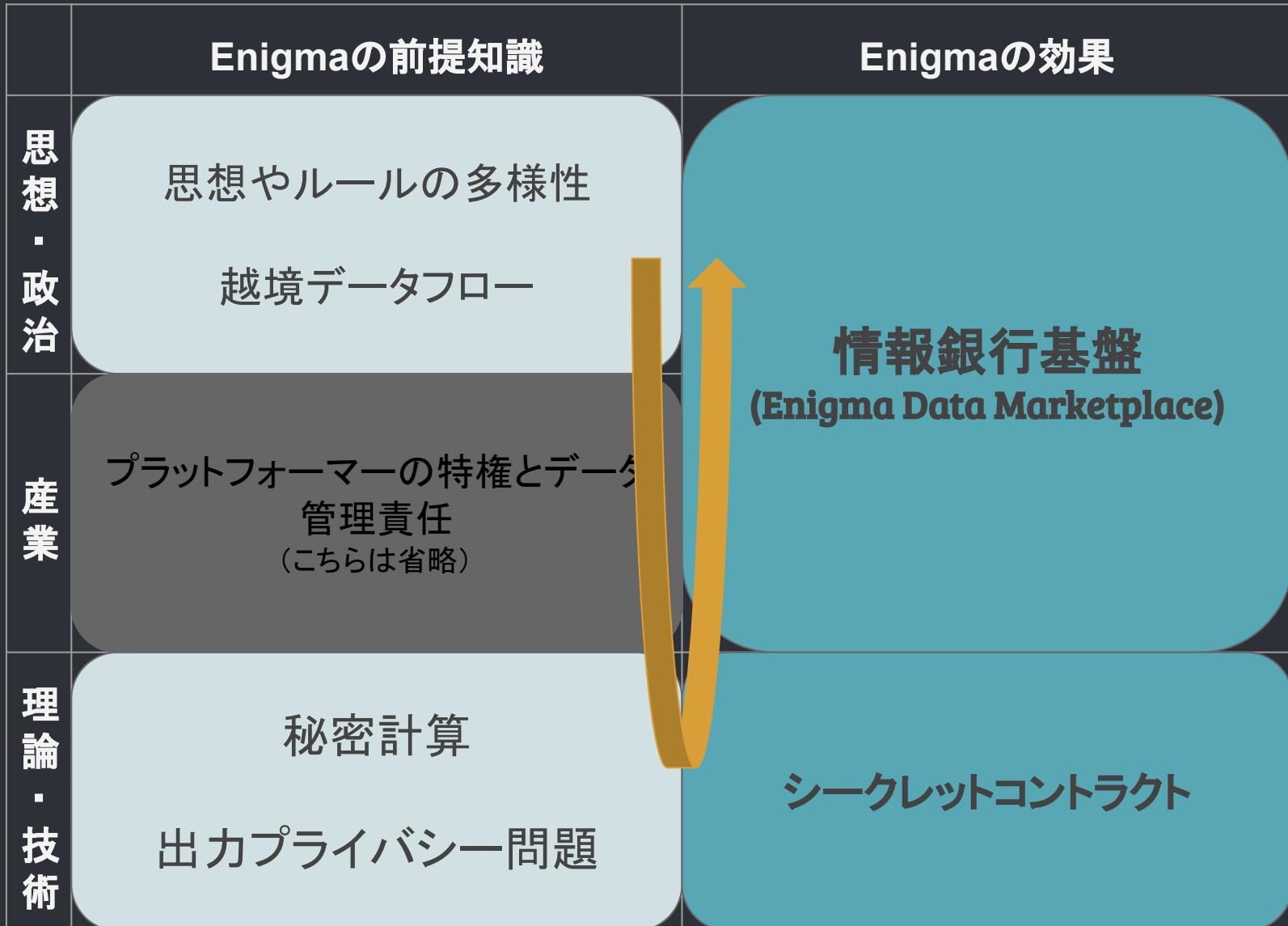




Context of Enigma

川口 将司 / Cho Jang Sa

- <この発表について>
発表内容をざっくりカテゴライズ
Enigmaを知るための前提知識 & Enigmaの未来展望を語る





<発表内容の整理>

各レイヤーを知るモチベーション

	Enigmaの前提知識	Enigmaの効果
思想 ・ 政治		<h1>Why</h1> <p>『なぜプライバシーを守らなければいけないの？』</p>
産業		<h1>What</h1> <p>『データを活用して何がしたいの？』 『データにどんな値打ちがあるの？』</p>
理論 ・ 技術		<h1>How</h1> <p>『どうやってデータを安全に活用すればいいの？』 『どうすれば、どのぐらいプライバシーを守れるの？』</p>

● もくじ



思想
・
政治

Enigmaの前提知識

思想やルールの多様性

越境データフロー

産業

プラットフォーマーの特権とデータ
管理責任
(こちらは省略)

理論
・
技術

秘密計算

出力プライバシー問題

Enigmaの効果

情報銀行基盤
(Enigma Data Marketplace)

シークレットコントラクト

<プライバシーの問題の本質>

モノと異なりデータの受け渡しは不可逆な行為である

- 受け渡し対象のコントロールを持ち主が取り戻すには.....

モノの場合



渡したら



返してもらえばいい

データの場合



渡したら



削除済みの
証明は困難



流出された
可能性も



システムの裏側でのデータの扱
われ方は証明困難である

● 取り戻せないプライバシーのデモンストレーション

モノの場合



渡したら



返してもらえばいい



...

データの場合



渡したら

記憶削除の困難

安○千代美たん
が好きなんです

とっても
センシティブ

流出の可能性



**不正確
リスク**



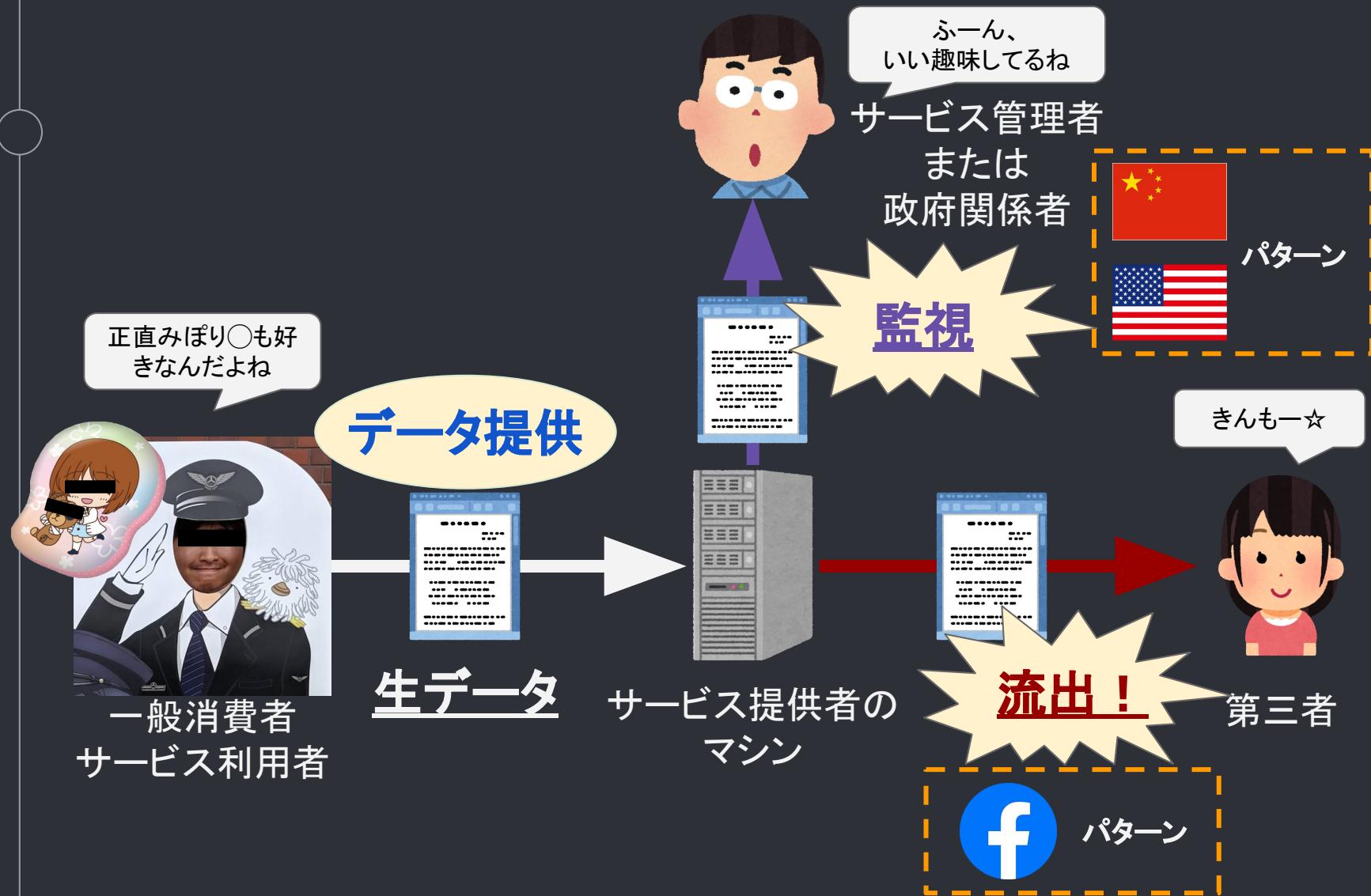
きんもー☆

チヨ○子
好きとな



ちょびww

- <データ管理責任者の権限濫用>
データの提供先は本当に信頼できる?
データの共有にはトラストフルな関係が求められる



<越境データフロー>

データ流通(地域間のデータ共有)の議論では『安全保障』『経済政策』『プライバシー』の3軸がバランス良く配慮されて欲しいが。

- 日本とアメリカはデータ活用推進主義(相対的にプライバシー軽視)
- EUはデータ保護主義(相対的にプライバシー偏重)
- 中国のデジタル保護主義は国内限定でデータ活用推進派、対外的にはデータ保護主義(国民のプライバシーは国家のもの)

地域(EU vs US)の例	EU	US
法源の考え方	法の遵守(法治国家)、制定法主義	法の発見(法の支配)、判例法主義
法体系・有効範囲	EU法、国内法	連邦法、州法(※1)
プライバシー権利の正式な宣言	欧洲人権条約8条(1950) → データ保護指令(1995) → 一般データ保護規則GDPR(2018)	法案のみ、議会では未成立(※2) → プライバシー権利章典法案(2015)
プライバシー保護の枠組み	GDPRなど	地域、分野ごとに異なる CCPA、COPPA、自主規制など
消費者の合意形成プロセス	Opt-in(※3)	Opt-out許容
プライバシー関連まとめ	EU法における“規則”で厳格なプライバシー保護、データ保護主義	包括的ではない(断片的な)法規制または自主規制、FTCによる監督・制裁

参考 : <https://www.meti.go.jp/metilib/report/H29FY/000807.pdf>

※1:州法が規定できない法規則の範囲はアメリカ合衆国憲法の第10条に定められる

※2: http://www.soumu.go.jp/main_content/000592528.pdf

※3: GDPR Recital32 “Silence, pre-ticked boxes or inactivity should not therefore constitute consent.”

- <マネジメントレベルの解決策>
異なる国や地域が要求するプライバシー保護の水準を満たすアプローチとして、ISO/IEC 29134:2017が標準規格として提唱されている

PbD/Privacy by Design実践手法の【PIA/Privacy Impact Assessment】により、パーソナルデータを扱うサービスの設計時点で、リスク要因分析および影響度の調査、リスク発生時の振る舞いを定める、という対策

現代の先進国のプライバシー保護観点を、最小公倍数的にカバーする。

※そのためか本文では『MUST』表現ではなく『SHOULD』表現が目立つ

期待される効果として：

- 消費者などステークホルダーとの信頼関係の構築
 - データ活用の目的や用途の説明をパブリックサマリにまとめる(※1)
- セキュリティ向上機会の提供
- 説明責任の全う
 - 他社へのデータ提供時にトレーサビリティを確保する、など
 - 指されても即答できる頼もしさ
- 開発コストの削減
 - システム開発が進行するほど、セキュリティ対策にかかる追加コストは膨張するため、事前の対策が有効である

※1:サンプルとしてUnited States Agency of International Developmentのパブリックサマリ

https://www.usaid.gov/sites/default/files/documents/1868/USAID.gov_PIA_Summary_20180627_v1D.pdf

● <プライバシーと分権>
しかし結局マネジメントレベルでの解決は少數組織の意図と能力に依存せざるを得ず、**権利侵害のリスクからは逃れられない**

あらゆるステークホルダーの要求を満たすデータ活用社会は、プライバシー侵害が可能な経路を極限まで無くすことのできるデータ活用基盤の上で実現される。

既存のスキームに足りないのは分権性

● <プライバシーと分権>

同時に『分権化され過ぎたプライバシー保護の仕組みが、社会にとって本当に心地の良いものか？』という議論も必要である

- 安全保障のための監視と、分権性の実現はぶつかり合う
- 犯罪・テロに都合の良いツールが実社会に受け入れられるだろうか？
- このあたりはアイデンティティと絡めた議論が必要と考える
 - 今回はスコープの範囲外（一人一）

● もくじ

	Enigmaの前提知識	Enigmaの効果
思想 ・ 政 治	思想やルールの多様性 越境データフロー	情報銀行基盤 (Enigma Data Marketplace)
産 業	プラットフォーマーの特権とデータ 管理責任 (こちらは省略)	
論 ・ 技 術	秘密計算 出力プライバシー問題	シークレットコントラクト

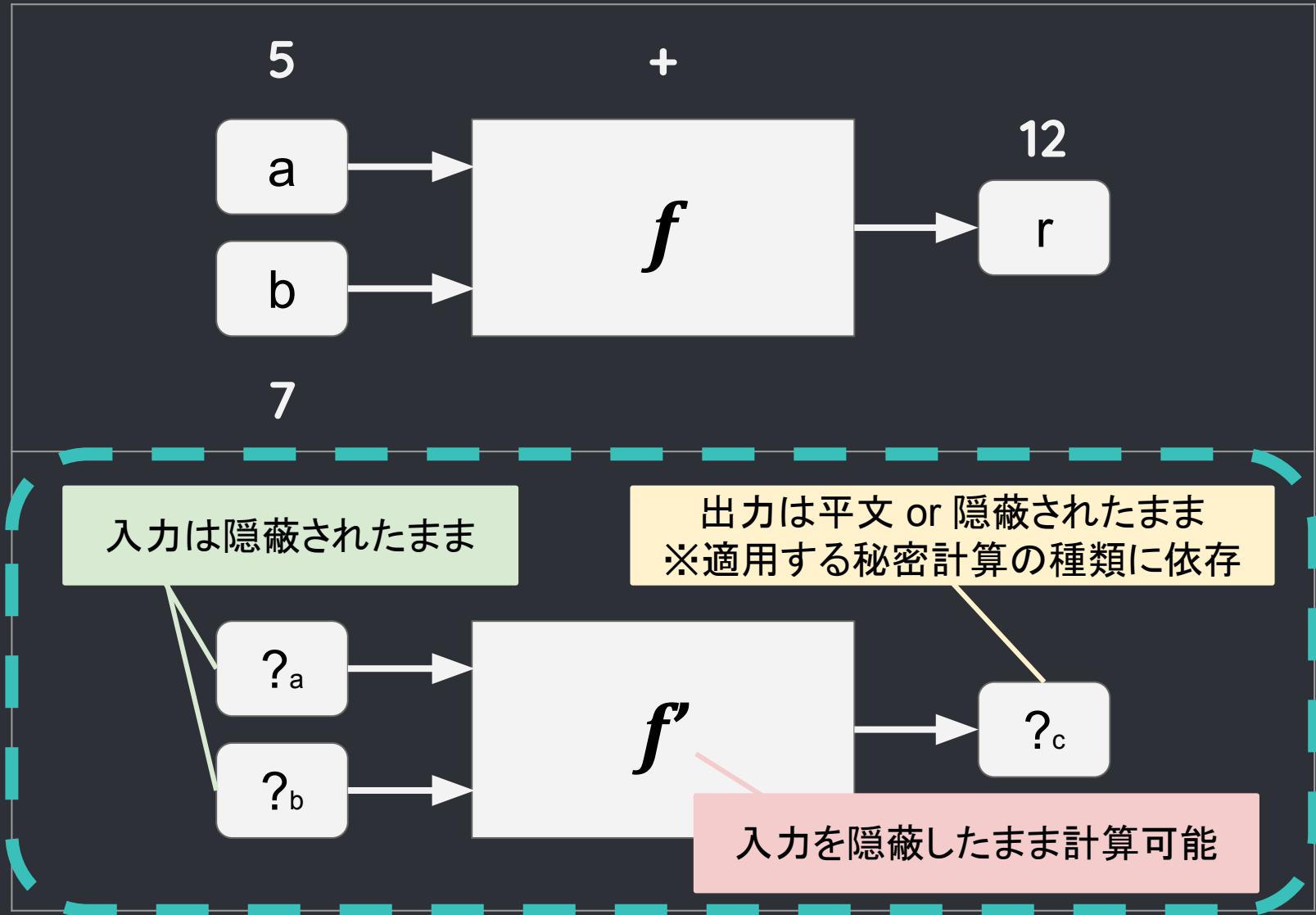
- もくじ



- 秘密計算とは
- 秘密分散ベースMPC vs TEE
- 出力プライバシー問題とは

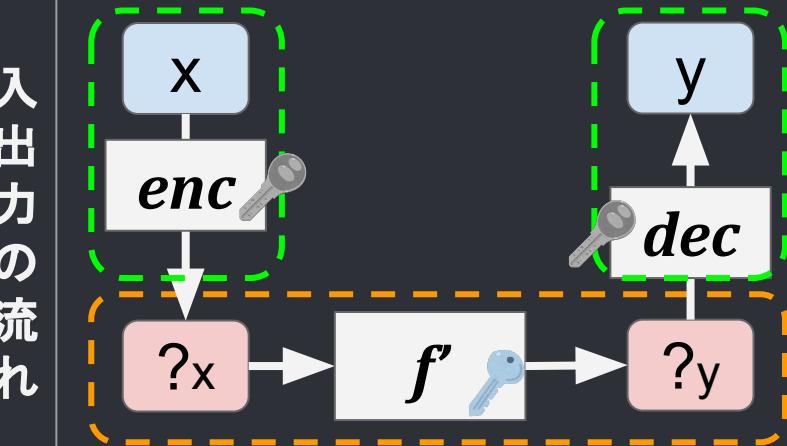
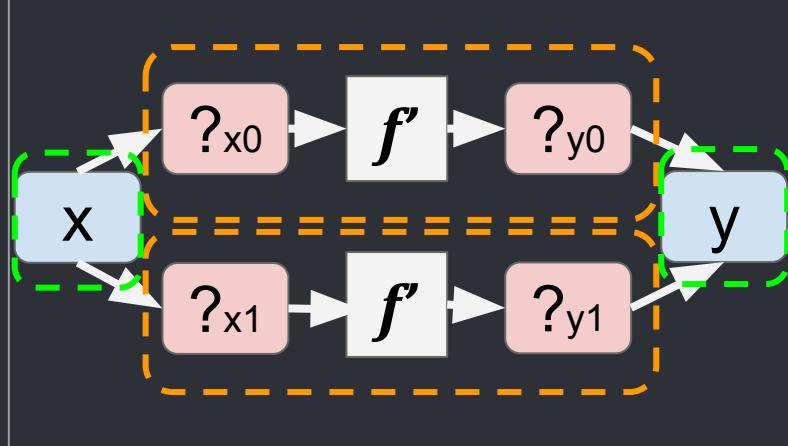
● <秘密計算とは>

秘密計算とは、入力を隠蔽したまま結果を導ける計算手法を指す
実用化するには出力プライバシー保護(後述)も別途必要である



<クラス毎に異なる秘密計算の特徴>

要件に応じた計算手法を選択し、その手法が与える制約やリスクを正確に理解した上で運用することが求められる

	完全準同型暗号(※)	秘密分散ベースMPC
メリット	<ul style="list-style-type: none">1. 同じ暗号文を異なる環境で活用可2. 通信回数が少ない	<ul style="list-style-type: none">1. 計算速度が速い(通信がネック)2. データサイズの膨張率が小さい
デメリット	<ul style="list-style-type: none">1. 計算速度が遅い2. データサイズの膨張率が高い3. 鍵を管理する必要がある	<ul style="list-style-type: none">1. 協力ノード間の通信回数が多い2. 管理者同士が談合しない複数の計算機が必要
入出力の流れ		

※近年”Multi-Key FHE”などの新しい手法が提唱されており、クラス毎に構成も異なる

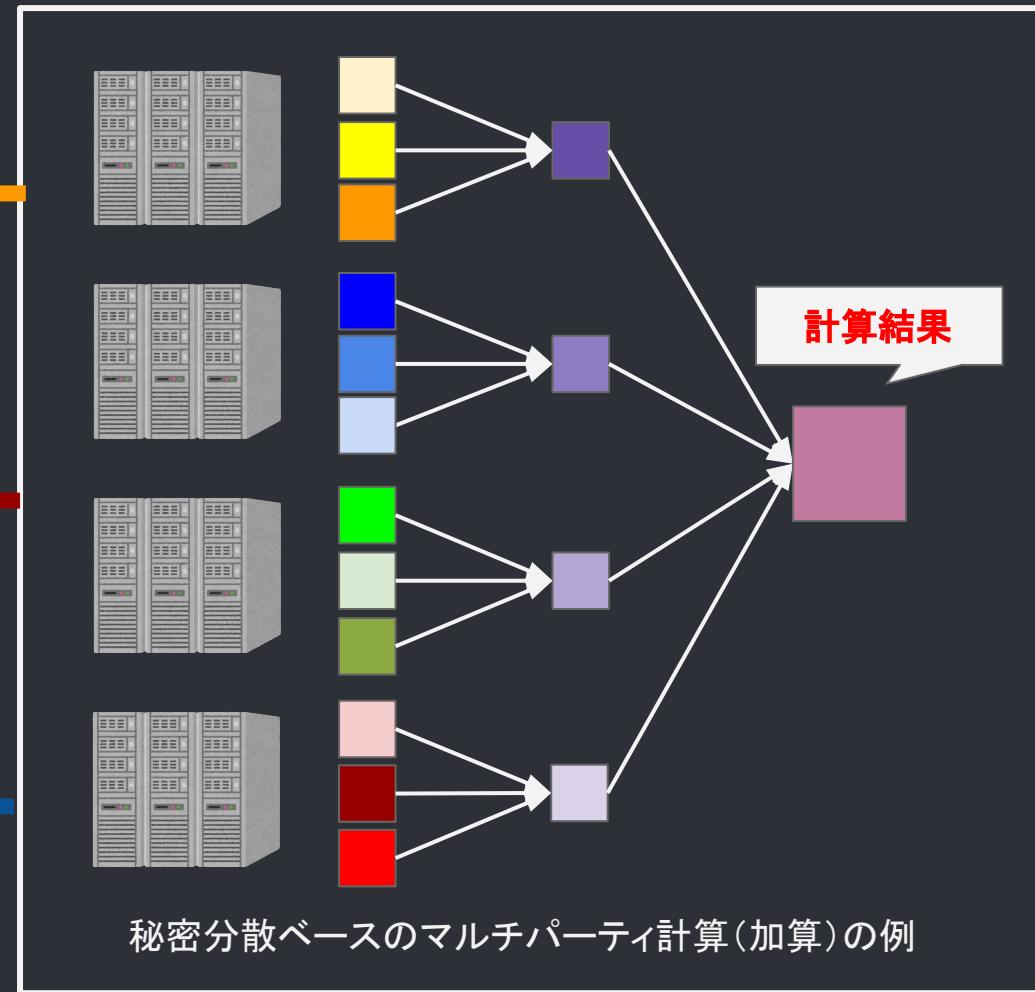
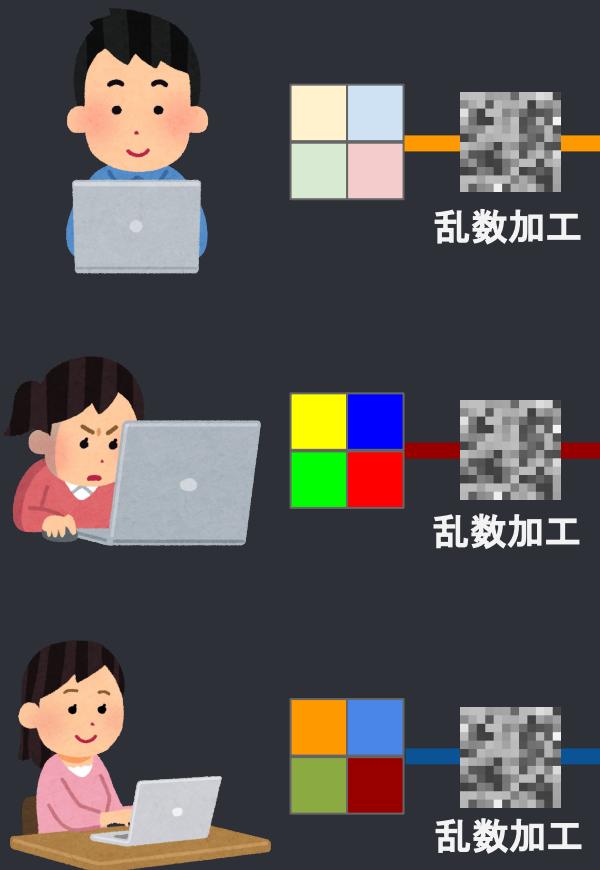
- もくじ

- 秘密計算とは
- 秘密分散ベースMPC vs TEE
- 出力プライバシー問題とは

<マルチパーティ計算(MPC/Multi-Party Computation) >

マルチパーティ計算とは

複数のノード上で秘密を分散して保有し、ノード間で協力しあって計算結果を導く手法のこと。主にGarbled Circuitと秘密分散の2種の実装がある



● <マルチパーティ計算(MPC/Multi-Party Computation) >
秘密分散ベースのマルチパーティ計算 ~加算編~

入出力の最大値を定義し、(最大値 + 1) = 法でmodulo計算を行う

秘密A =
 $(A_0 + A_1) \% \text{法}$



$A_0 = \text{乱数A}$



$c_0 = A_0 + B_0$

秘密B =
 $(B_0 + B_1) \% \text{法}$



$B_0 = \text{乱数B}$

$r = (c_0 + c_1) \% \text{法}$

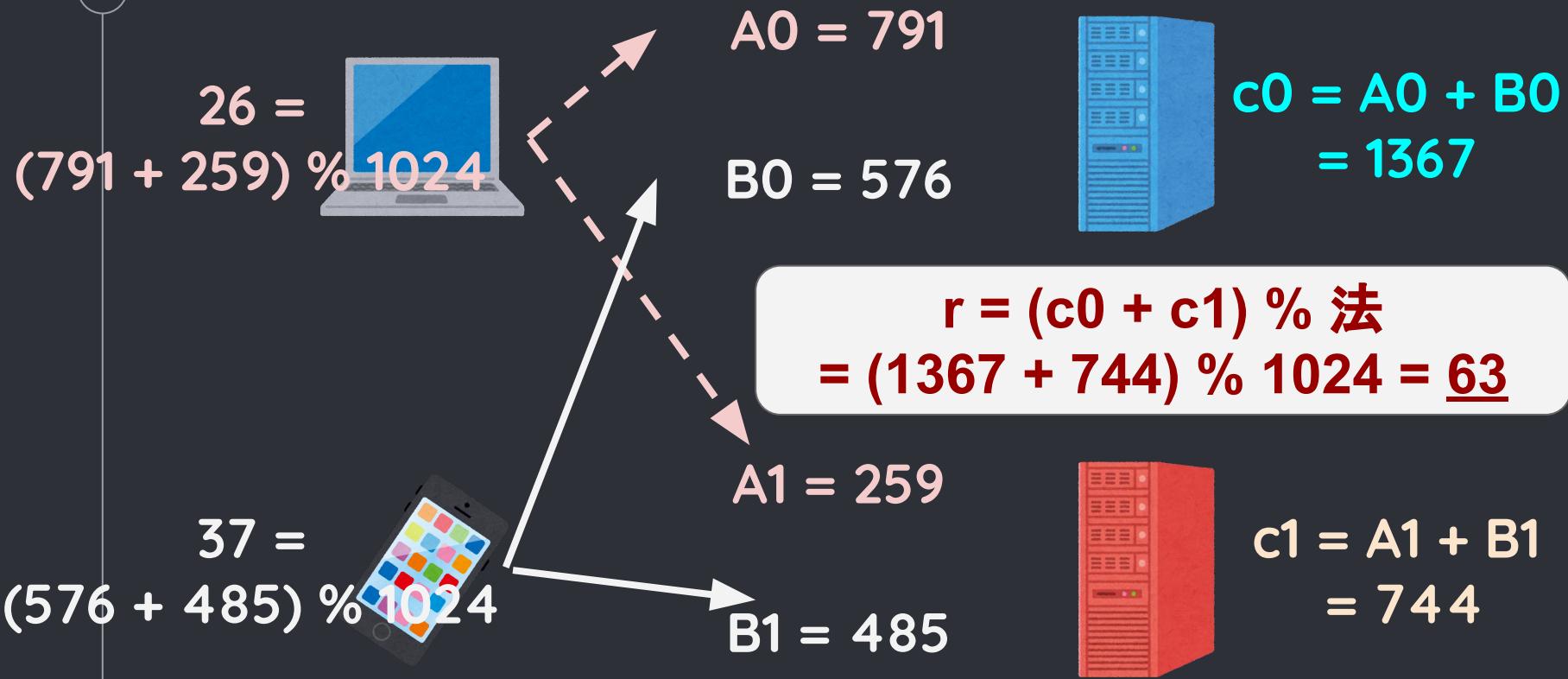
$A_1 =$
 $(\text{秘密A} - \text{乱数A}) \% \text{法}$



$c_1 = A_1 + B_1$

$B_1 =$
 $(\text{秘密B} - \text{乱数B}) \% \text{法}$

● <マルチパーティ計算(MPC/Multi-Party Computation) >
秘密分散ベースのマルチパーティ計算 ~加算編~



乗算の方が計算量・通信回数ともに多い
※興味ある人向けに処理の流れと検算を参考資料に掲載した

- もくじ

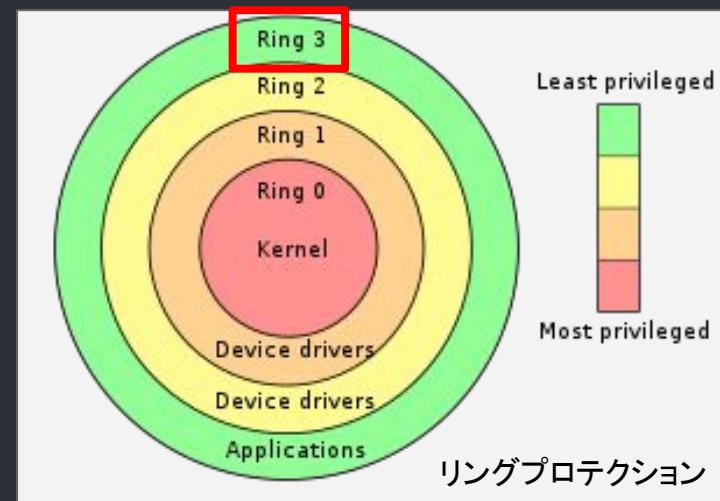
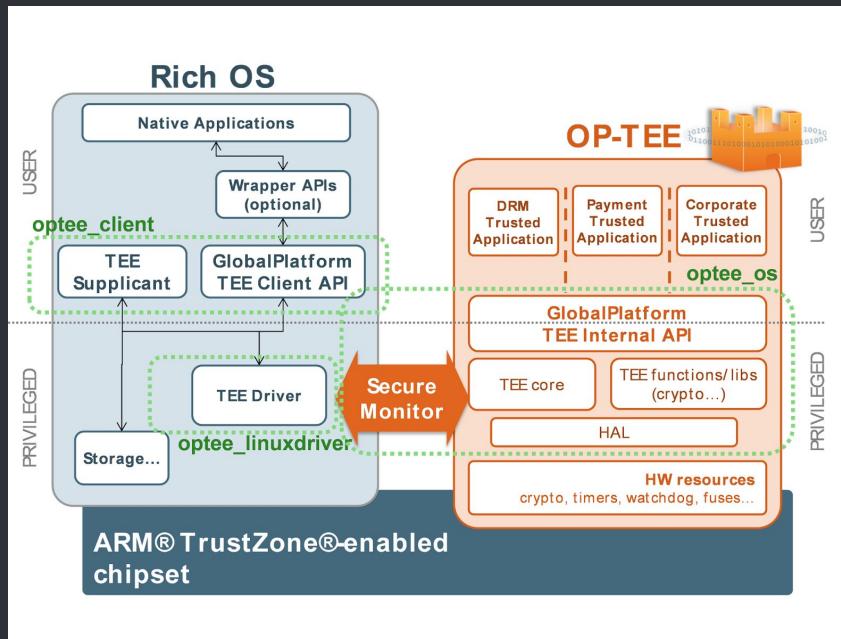
- 秘密計算とは
- 秘密分散ベースMPC vs TEE
- 出力プライバシー問題とは

<秘密分散ベース MPC vs TEE>

耐タンパ性に依存したハードウェアによる秘密計算スキーム

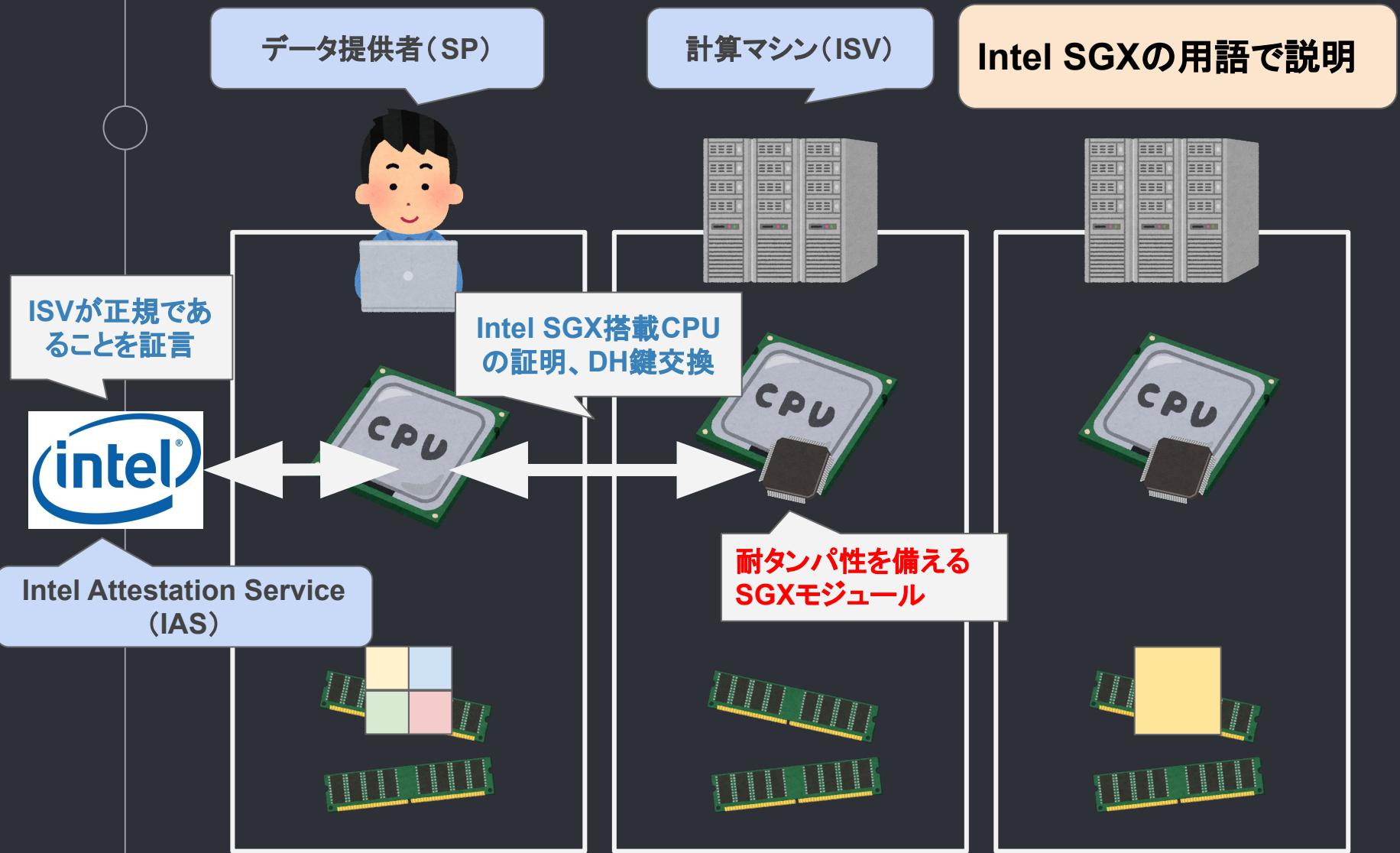
※このようなハードウェアに依存したスキームは 秘密計算の分野に数え上げないことが多い

- TEE(Trusted Execution Environment)
 - 耐タンパ性のあるモジュールで計算機の環境を『安全な領域』と『通常の領域』に切り分けるアプローチ
 - アプリケーションやOS、ミドルウェアとの相互アクセスを絞り込むことで、データ保護を図ろうというもの
- 多くの秘密計算と比較してパフォーマンス面で圧倒的に優れるが、サイドチャネル攻撃や分権性に課題を抱えている



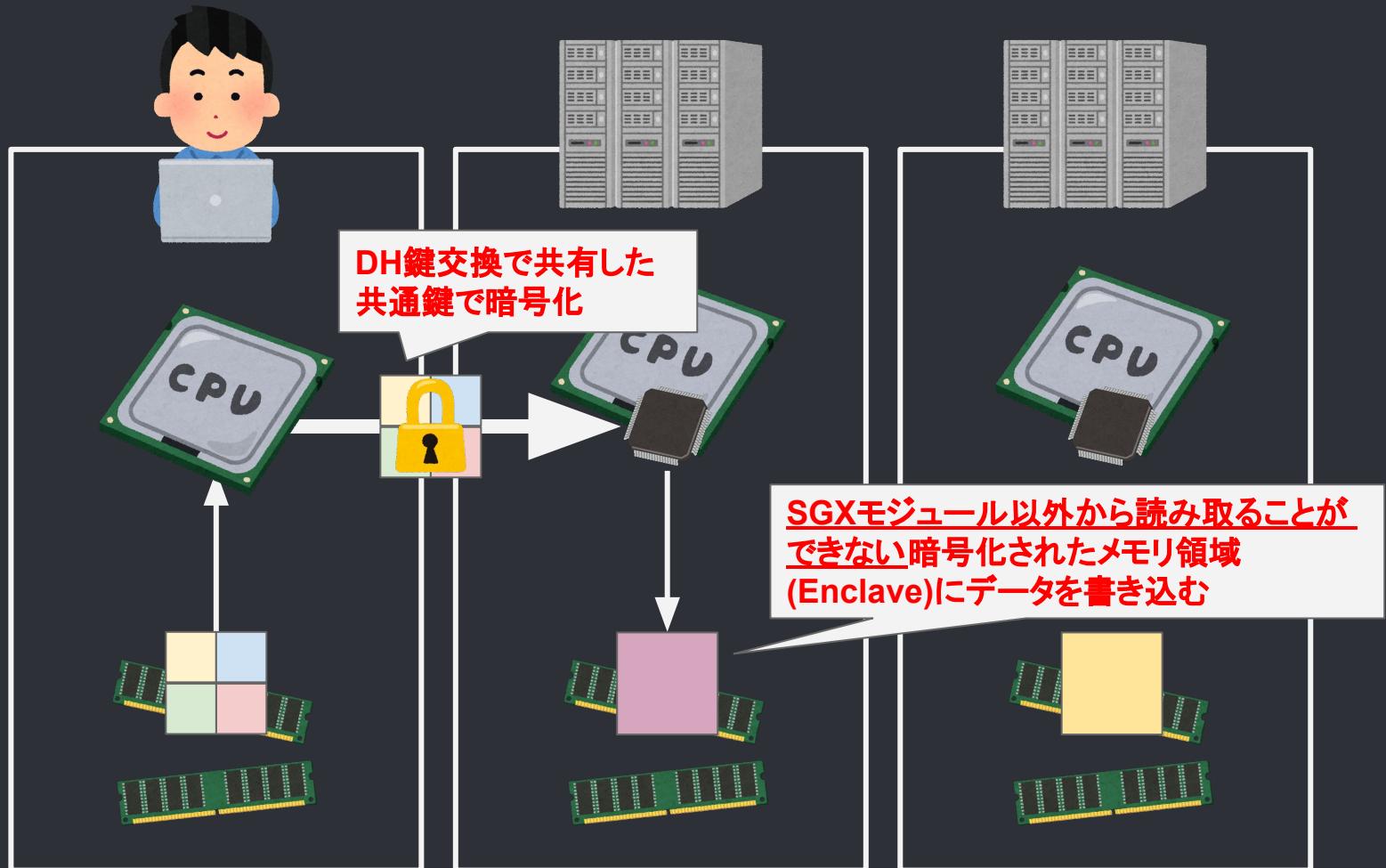
<秘密計算の種類 番外一TEE/Trusted Execution Environment>

TEEは計算機内において耐タンパ性のあるモジュールだけが復号可能な
Enclaveという暗号化済みメモリ領域を扱う



<秘密計算の種類 番外一TEE/Trusted Execution Environment>

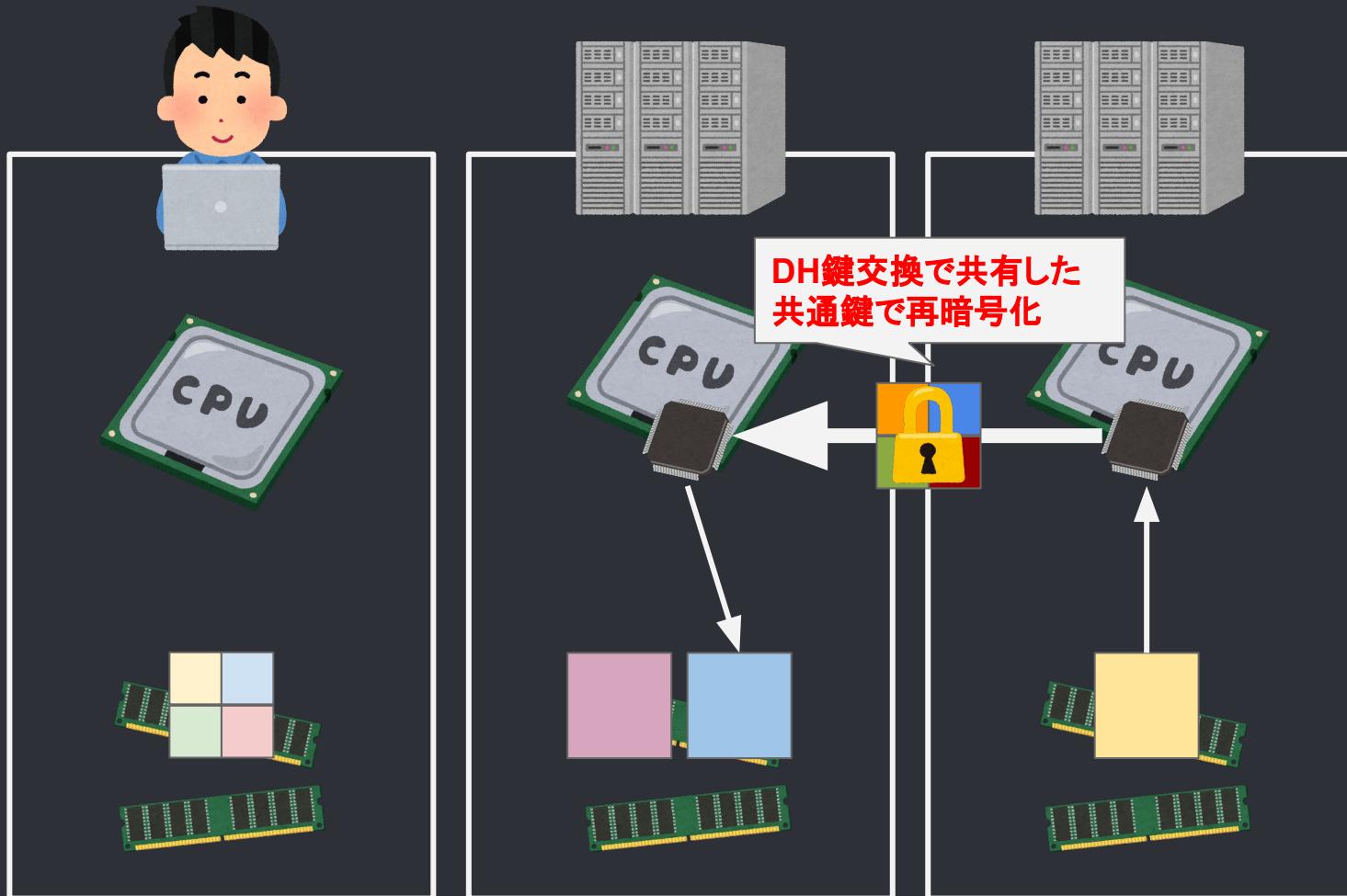
クライアントとサーバ双方のマシンがTEEに参加可能な正規のCPUであることを証明(Remote Attestation)した上でデータを送る



● <秘密計算の種類 番外一TEE/Trusted Execution Environment>

同じスキームで他のマシン同士の組み合わせでもデータを送ることができる

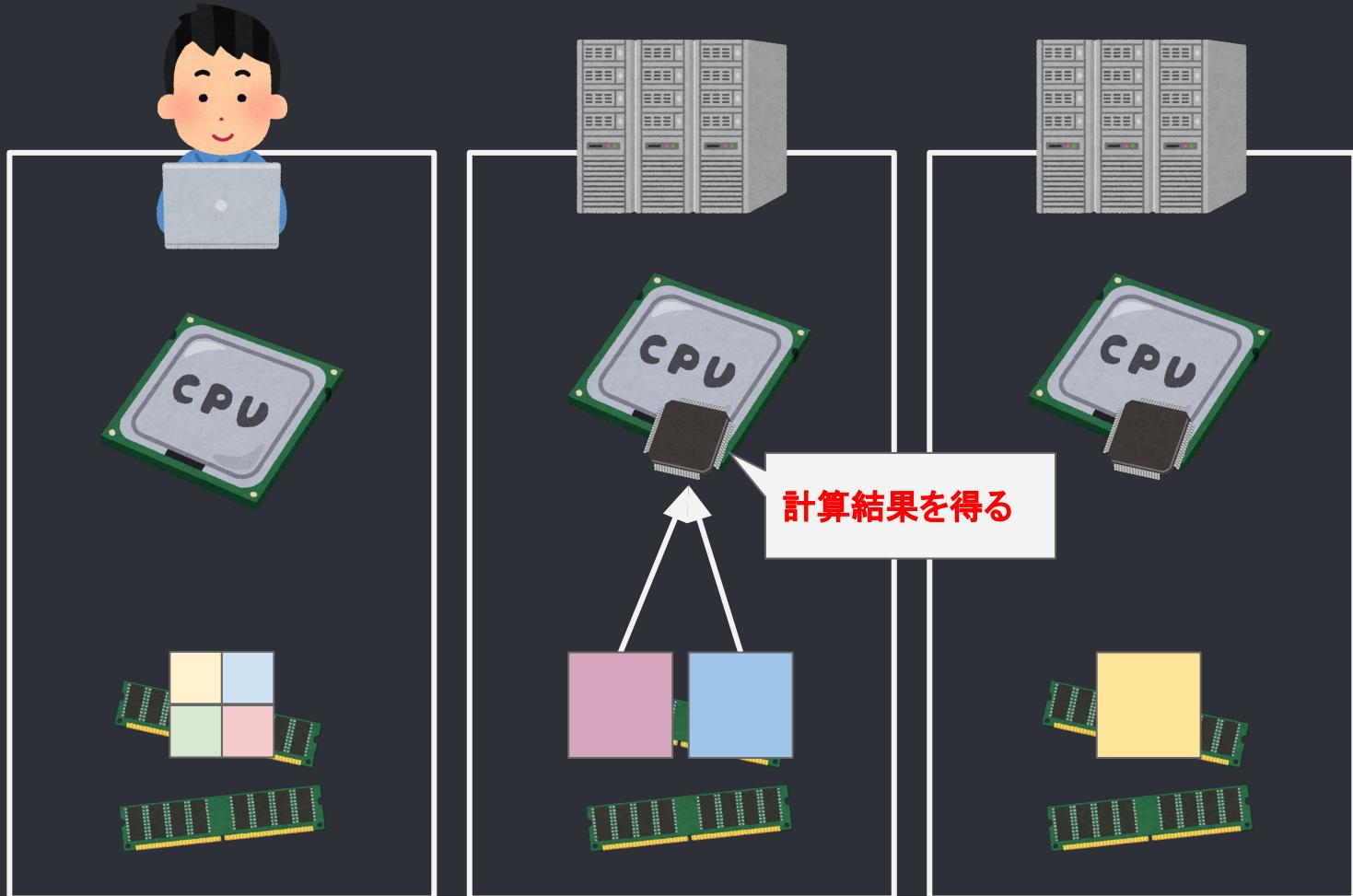
★  と  を引数とした計算をする時



<秘密計算の種類 番外一TEE/Trusted Execution Environment>

EnclaveはCPUに読み込まれるときに耐タンパ性のあるモジュールで復号されるので、計算を実行することが可能になる

★  と  を引数とした計算をする時



● <秘密分散ベース MPC vs TEE>

TEEの課題と展望

- サイドチャネル攻撃を受けるリスクを根本解決することは難しい
 - 一方で「サイドチャネル攻撃の再現には膨大なコストがかかるから大丈夫」と主張する人たちもいる
- Intel SGXの場合、Intelの認証スキームに依存するため、Intelを信頼することが大前提となる → 分権性に課題
- Proprietaryなハードウェアは内部の振る舞いが不透明である
 - オープンソースハードウェアの『RISC-V』に注目
 - ライセンス料／ロイヤリティなしにISAを使うことができ、設計開発したハードウェアのIPは自社のものにできる
 - モジュラーISAにより柔軟性と実行効率の高いコアの開発
 - <https://riscv.org/>

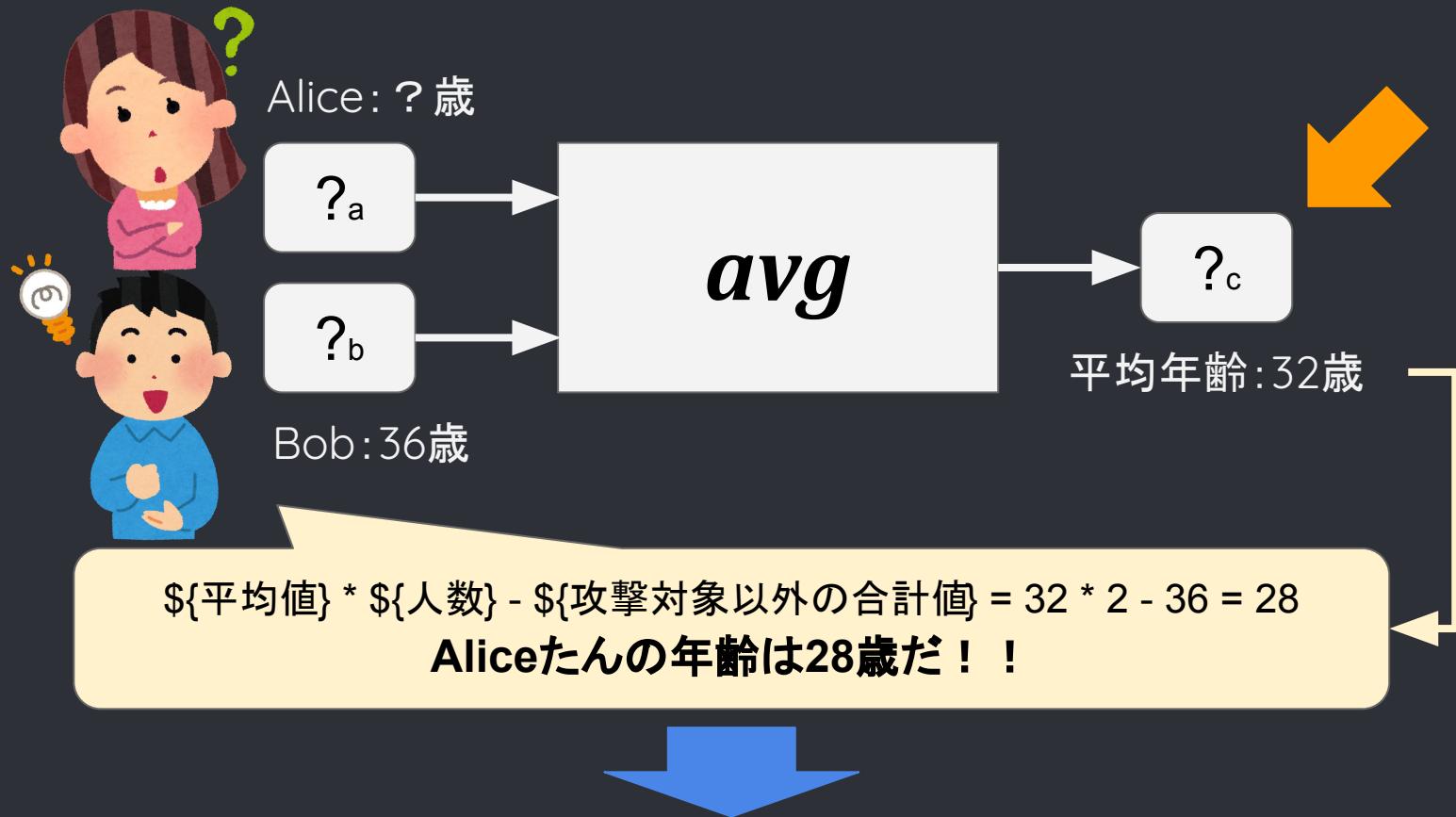
- もくじ

- 秘密計算とは
- 秘密分散ベースMPC vs TEE
- 出力プライバシー問題とは

＜出力プライバシー問題とは＞

秘密計算で『入力』を隠蔽したまま計算できることはわかった。

一方の『出力』は野放しで良いのだろうか？



出力からプライバシーが侵される可能性 → 出力プライバシー問題

上記の例のほか、クエリ結果から従属性に値を割り出したり、比較演算でとりうる値を絞り込んだり、微小なサンプルと合算値から値の大きなサンプルの値を割り出したり(n-k占有)、さまざまな脅威が考えられる

● <出力プライバシー保護の強度の定量表現>
対策の例: 差分プライバシー(ϵ -Differential Privacy)の紹介

前スライドの例のように、攻撃者は出力結果から入力値(データベース)のうち1要素ずつ明らかにする攻撃を試みる。

これに対し、出力結果に統計的乱雑化(ラプラスノイズ)を加えることで、上記の攻撃手法を妨害する差分プライバシーのモデルを紹介する。

この手法の要点は入力値のデータベース内に1要素の有る無しが計算結果に与える差分(=その1要素が計算結果に与えるユニークさ)を小さくできるノイズをえて隠そうというもの。

このノイズ付加後の確率(密度)の差異を ϵ で定量的に表現する。裏を返せば ϵ が0に近づくほど、1要素を足した結果の差異が失われていく。

平均値の計算の例では、データベースのサイズが大きいほど、1要素が計算結果に与える影響は小さい。つまりデータベースのサイズが大きいほど、大きなノイズをえて計算結果を歪めずに済む。



＜出力プライバシー保護の強度の定量表現＞ ϵ はどのような値か

高々1要素だけ異なる2つのデータベースD1、D2に対し、以下を満たす関数Aは ϵ -Differential Privacyを満たす。

$$\begin{aligned} \star \Pr [A(D1) \in S] &\leq \exp(\epsilon) * \Pr[A(D2) \in S] \\ \rightarrow \log \Pr [A(D1) \in S] &\leq \log \exp(\epsilon) * \Pr[A(D2) \in S] \\ \rightarrow \log \Pr [A(D1) \in S] &\leq \log \exp(\epsilon) + \log \Pr[A(D2) \in S] \\ \rightarrow \log \Pr [A(D1) \in S] &\leq \epsilon + \log \Pr[A(D2) \in S] \\ \rightarrow |\log \Pr [A(D1) \in S] - \log \Pr[A(D2) \in S]| &\leq \epsilon \end{aligned}$$

ただし、

A:ノイズを付加するプライバシー機構の関数

S:Aの出力が取りうる値域のサブセット

Pr:確率(密度)

※ちなみに上記式の右辺に微小な確率(密度)δを加えて、 (ϵ, δ) -Differential Privacyとするバリエーションも存在する。やはりδも小さいほどプライバシーの強度は高い。

“

ここまでがEnigmaを知るための下地
ここからがEnigmaのおはなし

● もくじ

	Enigmaの前提知識	Enigmaの効果
思想 ・ 政治	思想やルールの多様性 越境データフロー	情報銀行基盤 (Enigma Data Marketplace)
産業	プラットフォーマーの特権とデータ 管理責任 (こちらは省略)	
理論 ・ 技術	秘密計算 出力プライバシー問題	シークレットコントラクト



Enigmaとは



Enigma [?]

- Enigmaとは



Enigma [sMPC]

- Enigmaとは



Enigma [TEE]

- Enigmaとは

Enigma [sMPC]

Enigma [TEE]

Enigma [?]

.....

...

- Enigmaとは



Enigma [sMPC]

≠

Enigma [TEE]

- Enigmaとは



Enigma [?]

<結論>

● Enigmaは秘密計算に**中立性**をもたらす仕組みである

- シークレットコントラクトは公開される
 - つまりデータに対する操作方法が公開される
- つまりデータにどのような操作がされるか、あらかじめ把握できる
 - 定義された処理が受け入れ可能か否か、出力プライバシーなど総合的に鑑みて、データ提供者自身が判断できる
- おまけにサービスのバイアス問題を解決する → 次スライド

＜中立性の問題＞

ひとびとはサービス／学習モデルの由来成分・制作工程を知らない
結果的にバイアスに晒される

- 商業的・政治的利害のために、不都合なデータは取り除かれた上で、そのサービス(アプリケーション／モデル)は提供されているかもしれない
- 以下に挙げるような情報を操作することで利害が生まれる
 - 製品の評価、レコメンド
 - 歴史・経緯・定義の説明
 - 価格情報・シグナル
 - ほか
- データ活用社会の発展と共に、この影響は今後さらに強まっていく

個人であれ、組織であれ、あらゆる情報を好き嫌いせず、公正公平に取り扱うことは難しい



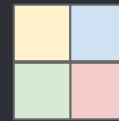
<Enigmaの概要>

Enigmaネットワークにおけるデータとプログラムの配置

★sMPCモードなEnigmaの環境



生データ



乱数加工

シークレットコントラクト

```
/** Simple HelloButton() method.  
 * version 1.0  
 * @author john doe <doe.j@example.com>  
 */  
HelloButton()  
{  
    JButton hello = new JButton("Hello, world!");  
    hello.addActionListener(new HelloButtonListener());  
  
    // use the JFrame type until support for  
    // new components is finished  
    JFrame frame = new JFrame("Hello Button");  
    Container pane = frame.getContentPane();  
    pane.add(hello);  
    frame.pack();  
    frame.setVisible(true);  
}  
  
/** Simple HelloButton() method.  
 * version 1.0  
 * @author john doe <doe.j@example.com>  
 */  
HelloButton()  
{  
    JButton hello = new JButton("Hello, world!");  
    hello.addActionListener(new HelloButtonListener());  
  
    // use the JFrame type until support for  
    // new components is finished  
    JFrame frame = new JFrame("Hello Button");  
    Container pane = frame.getContentPane();  
    pane.add(hello);  
    frame.pack();  
    frame.setVisible(true);  
}  
  
/** Simple HelloButton() method.  
 * version 1.0  
 * @author john doe <doe.j@example.com>  
 */  
HelloButton()  
{  
    JButton hello = new JButton("Hello, world!");  
    hello.addActionListener(new HelloButtonListener());  
  
    // use the JFrame type until support for  
    // new components is finished  
    JFrame frame = new JFrame("Hello Button");  
    Container pane = frame.getContentPane();  
    pane.add(hello);  
    frame.pack();  
    frame.setVisible(true);  
}  
  
/** Simple HelloButton() method.  
 * version 1.0  
 * @author john doe <doe.j@example.com>  
 */  
HelloButton()  
{  
    JButton hello = new JButton("Hello, world!");  
    hello.addActionListener(new HelloButtonListener());  
  
    // use the JFrame type until support for  
    // new components is finished  
    JFrame frame = new JFrame("Hello Button");  
    Container pane = frame.getContentPane();  
    pane.add(hello);  
    frame.pack();  
    frame.setVisible(true);  
}
```

許可

許可



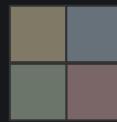
<Enigmaの概要>

Enigmaネットワークにおけるデータとプログラムの配置

★sMPCモードなEnigmaの環境



生データ



シーケレットコントラクト

```
/** Simple HelloButton() method.  
 * version 1.0  
 * @author john doe <doe.j@example.com>  
 */  
HelloButton()  
{  
    JButton hello = new JButton("Hello, world!");  
    hello.addActionListener(new HelloActionListener());  
    // use the JFrame type until support for  
    // new component is finished  
    JFrame frame = new JFrame();  
    JButton hello = new JButton("Hello, world!");  
    hello.addActionListener(new HelloActionListener());  
    frame.getContentPane().add(hello);  
    frame.pack();  
    frame.show();  
}  
  
/** Simple HelloButton() method.  
 * version 1.0  
 * @author john doe <doe.j@example.com>  
 */  
HelloButton()  
{  
    JButton hello = new JButton("Hello, world!");  
    hello.addActionListener(new HelloActionListener());  
    // use the JFrame type until support for  
    // new component is finished  
    JFrame frame = new JFrame();  
    JButton hello = new JButton("Hello, world!");  
    hello.addActionListener(new HelloActionListener());  
    frame.getContentPane().add(hello);  
    frame.pack();  
    frame.show();  
}  
  
/** Simple HelloButton() method.  
 * version 1.0  
 * @author john doe <doe.j@example.com>  
 */  
HelloButton()  
{  
    JButton hello = new JButton("Hello, world!");  
    hello.addActionListener(new HelloActionListener());  
    // use the JFrame type until support for  
    // new component is finished  
    JFrame frame = new JFrame();  
    JButton hello = new JButton("Hello, world!");  
    hello.addActionListener(new HelloActionListener());  
    frame.getContentPane().add(hello);  
    frame.pack();  
    frame.show();  
}
```

許可

許可

オンチェーン (Layer1)

- ・ブロックチェーン上に刻まれる情報で、改ざんが困難
- ・無対策では刻まれる情報がパブリック
- ・一般的に書き込みが低速である



● <Enigmaの概要>
Enigmaのオンチェーンに刻まれる情報

シークレットコントラクトの
Hash値、TaskID



$h(f)$

計算・取引の履歴



署名



計算執行元
アドレス



手数料

データの所在とアクセス権
(トラッカー的役割)



計算
OK

復号
OK

NG

e.t.c. (実装に依存)



保有残高



難易度

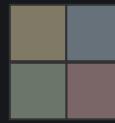
<Enigmaの概要>

Enigmaネットワークにおけるデータとプログラムの配置

★sMPCモードなEnigmaの環境



生データ



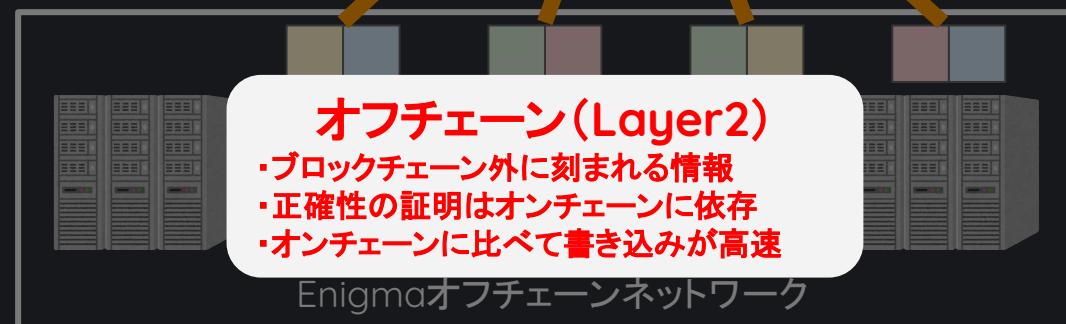
乱数加工

シークレットコントラクト

```
/** Simple HelloButton() method.  
 * version 1.0  
 * @author john doe <joe.j@example.com>  
 */  
HelloButton()  
{  
    JButton hello = new JButton("Hello, world");  
    hello.addActionListener(new HelloButtonListener());  
    // use the JFrame type until support for  
    // new component is finished  
    JFrame frame = new JFrame();  
    JButton hello = new JButton("Hello, world");  
    hello.addActionListener(new HelloButtonListener());  
    frame.getContentPane().add(hello);  
    frame.pack();  
    frame.show();  
}  
  
/** Simple HelloButton() method.  
 * version 1.0  
 * @author john doe <joe.j@example.com>  
 */  
HelloButton()  
{  
    JButton hello = new JButton("Hello, world");  
    hello.addActionListener(new HelloButtonListener());  
    // use the JFrame type until support for  
    // new component is finished  
    JFrame frame = new JFrame();  
    JButton hello = new JButton("Hello, world");  
    hello.addActionListener(new HelloButtonListener());  
    frame.getContentPane().add(hello);  
    frame.pack();  
    frame.show();  
}  
  
/** Simple HelloButton() method.  
 * version 1.0  
 * @author john doe <joe.j@example.com>  
 */  
HelloButton()  
{  
    JButton hello = new JButton("Hello, world");  
    hello.addActionListener(new HelloButtonListener());  
    // use the JFrame type until support for  
    // new component is finished  
    JFrame frame = new JFrame();  
    JButton hello = new JButton("Hello, world");  
    hello.addActionListener(new HelloButtonListener());  
    frame.getContentPane().add(hello);  
    frame.pack();  
    frame.show();  
}
```

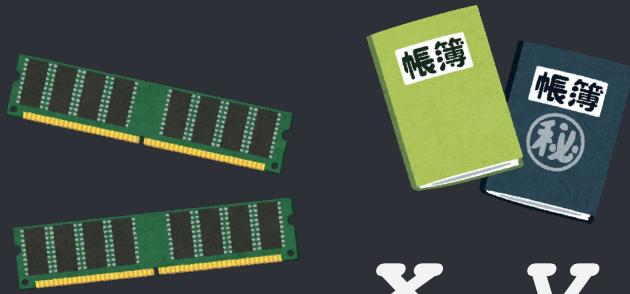
許可

許可



● <Enigmaの概要>
Enigmaのオフチェーンに刻まれる情報

計算対象データ



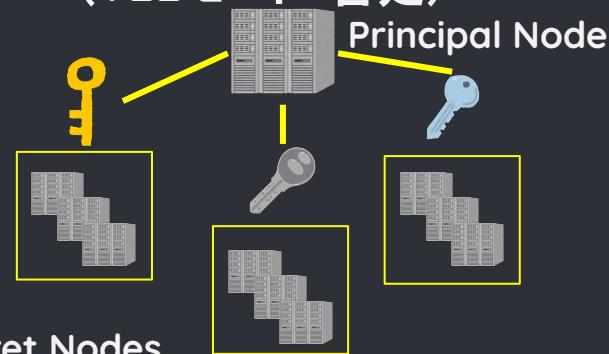
シークレットコントラクト



f

グループの共有鍵

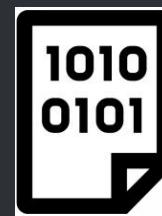
(TEEモード・暫定)



e.t.c. (一部暫定)



s3/IPFS等の
リンク情報



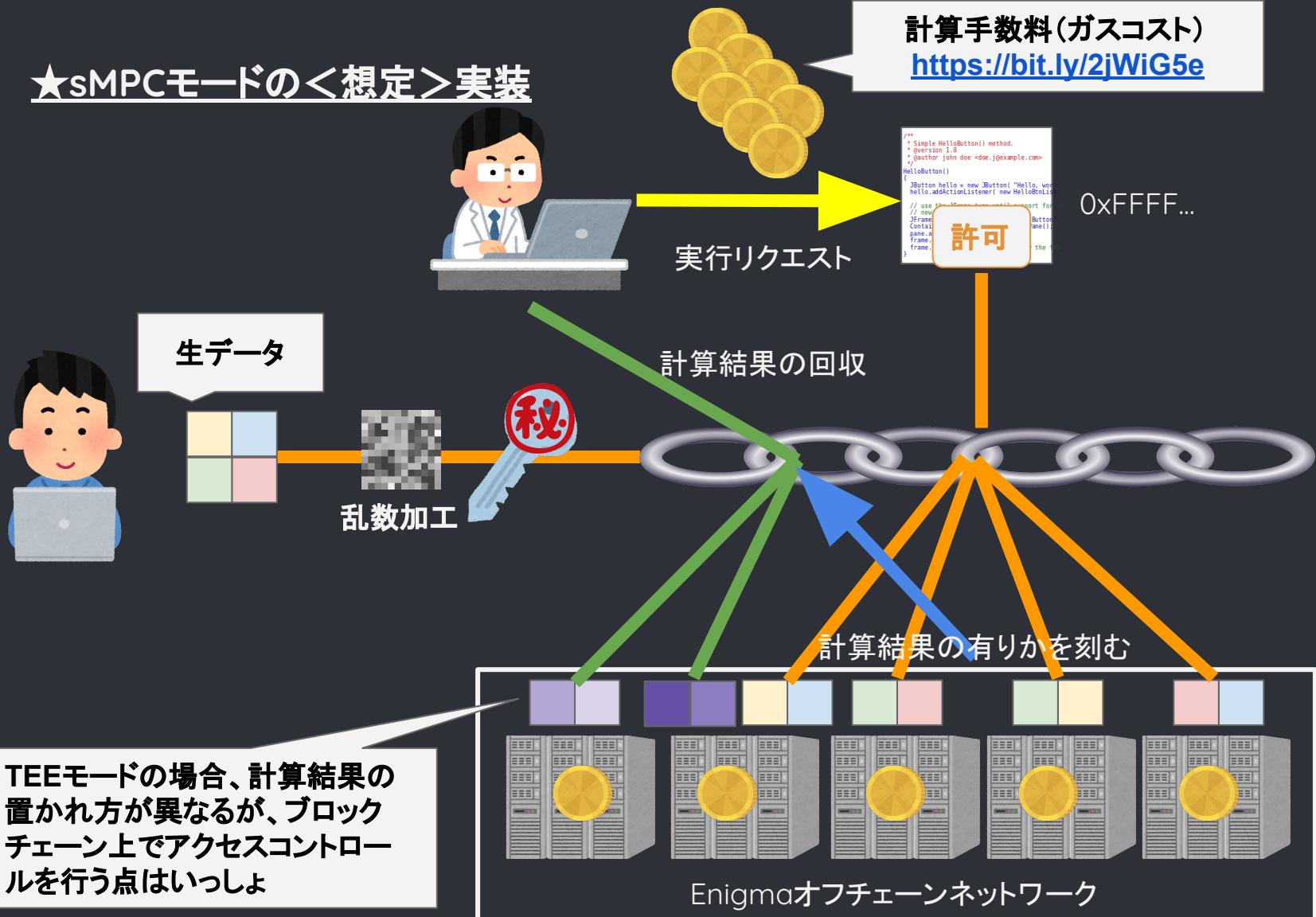
コントラクト毎の
ワーカー一覧

コントラクトのバイトコード
あるいはその参照

<Enigmaの概要>

Enigmaネットワーク上の秘密計算、シークレットコントラクトの執行

★sMPCモードの<想定>実装



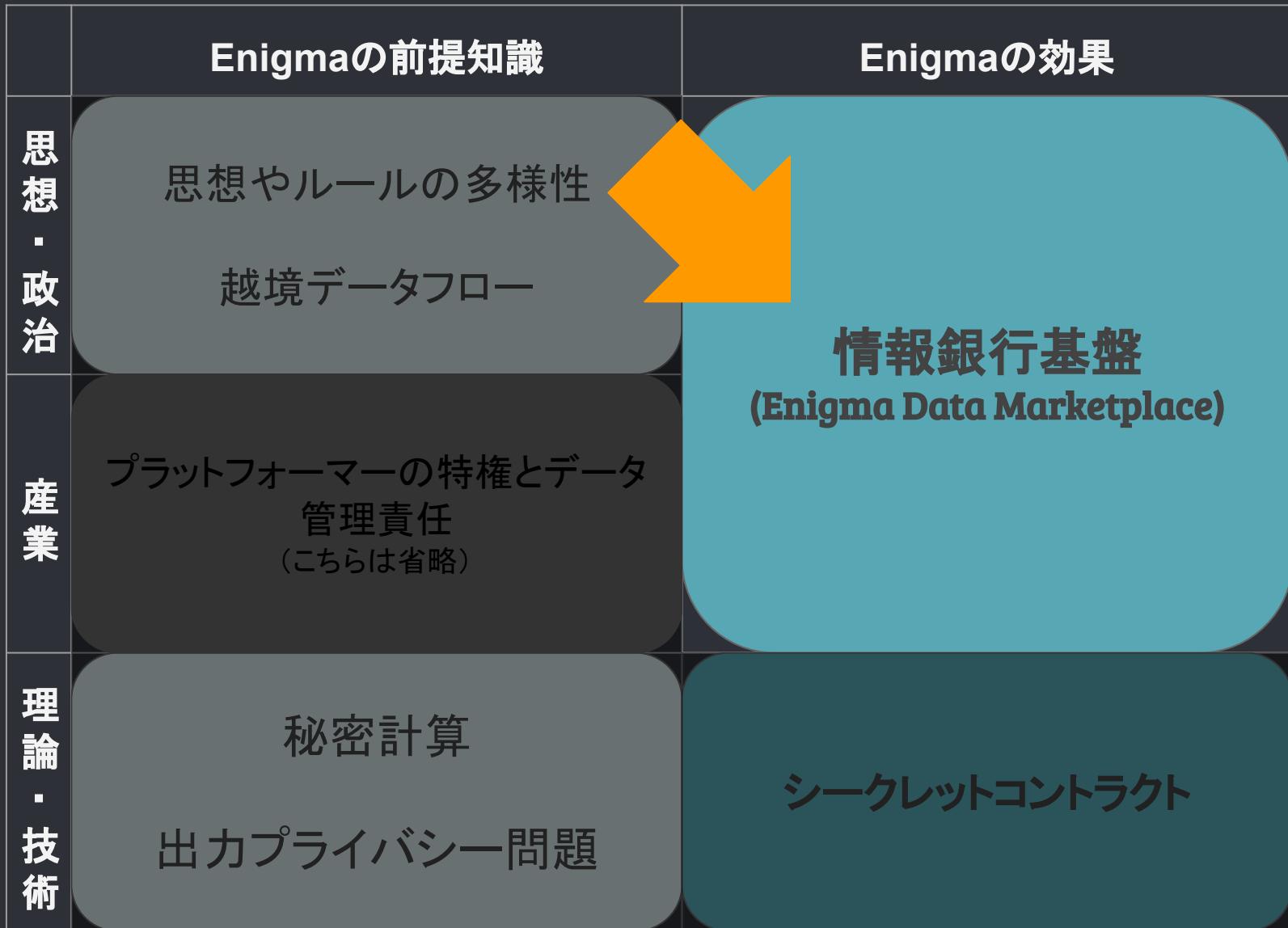


<シークレットコントラクト>
シークレットコントラクトのサンプル



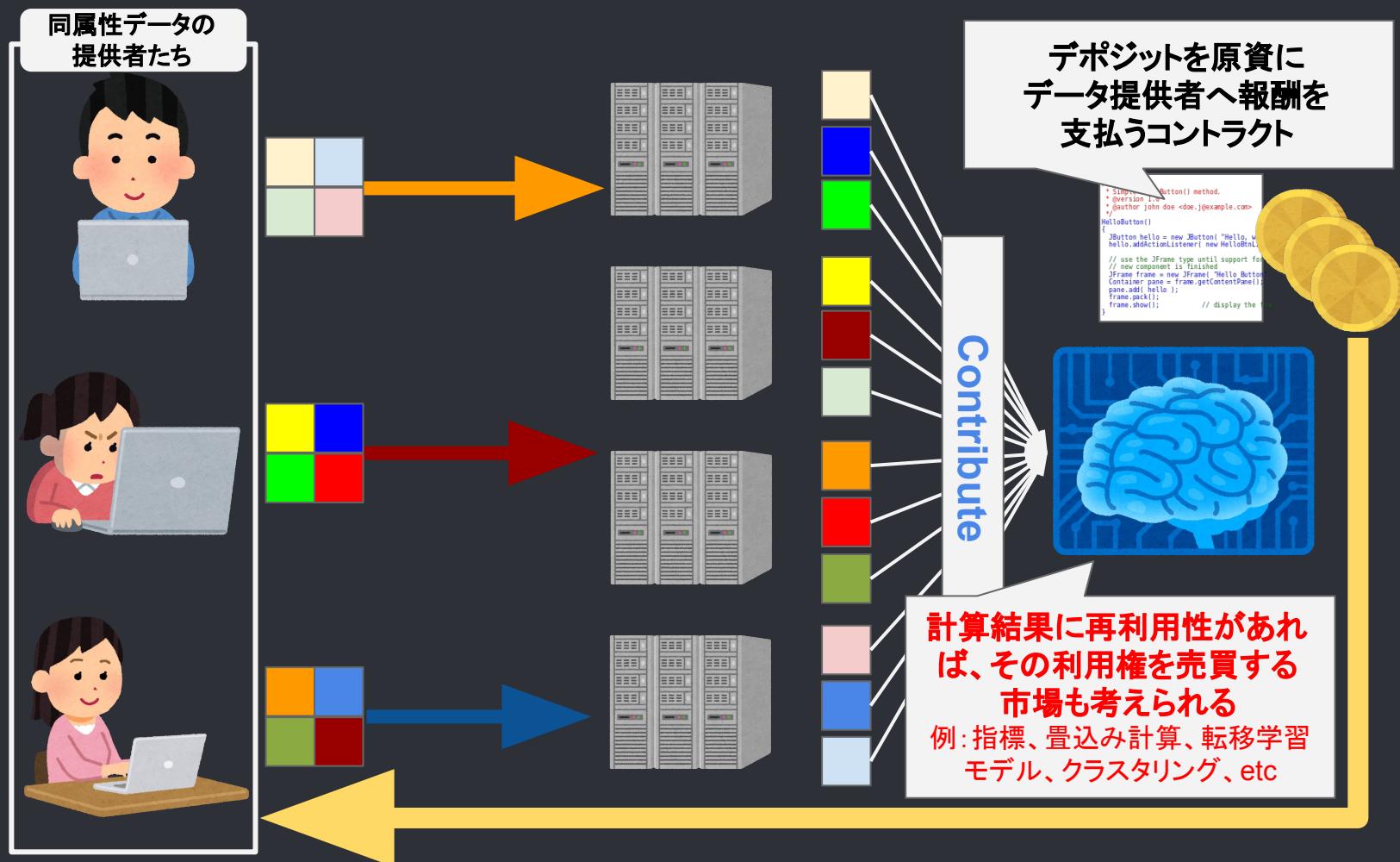
<https://enigma.co/discovery-documentation/SecretContractExamples/>

● もくじ



● <Enigmaの代表的なユースケース>
Enigma Data Marketplace(情報銀行基盤)

「データ提供 -> データ活用 -> 値値の還元」というエコシステムを実現できるプラットフォームを提供する。



● <国家が主導するデータ活用機会の拡大>
データ活用の自由化に向けて協調路線を歩んでいる日本と米国
個人の権利よりデータ活用機会を優先、データ保護主義への対抗

各政府の視点では、中国のデジタル保護主義やGAFA等のデータ霸権主義に対抗することが1つのテーマといえる。人口(≒データ量)からして当然、多国間・業界間の協力が必要不可欠である。

今年1月の世界経済フォーラム年次総会(ダボス会議)で、国境を越えたデータ流通の制度を提言をしていく日本国政府の方針が示された。

Data Free Flow with Trust/DFFT

6月、『大阪トラック』と銘打ちG20大阪サミットで打ち出されたが、EUや中国はもちろん、インドやインドネシアにも賛同を得られず。(※1)

引き続き、次回WTO閣僚会議における反応に注目したい。

いずれにせよ真の意味で『with Trust』であるためには、まずステークホルダー同士が納得できるデータ共有基盤が必要不可欠なのではないか。

Enigmaが提供する計算基盤は、その選択肢のひとつに数えられる。

※1 : <https://www.sankei.com/west/news/190628/wst1906280029-n1.html>

＜情報銀行の役割＞

『情報銀行』はあくまで『銀行』なんだけど、

『情報商社』になってないよね??

預けたプライバシーは本当に取り戻せるのだろうか？

モノの場合



渡したら

返してもらえばいい



...

データの場合



渡したら

記憶削除の困難

安○千代美たん
が好きなんです

とっても
センシティブ



不正確
リスク

きんもー☆

チョ○子
好きとな



ちょびww

流出の可能性

● <まとめ>

Enigmaはデータ活用社会の実現に向けた種々の課題を解決する

Enigmaは以下の特徴を備えることで、

地域ごとのルール、個々人の意思を尊重したデータ活用基盤を提供することができる

- 秘密計算による入力プライバシー保護
- データに対する操作内容を公然にする
- データ提供者がデータのアクセス権を支配できる
 - 積極的プライバシーを技術で公平公正に成就させる

またEnigmaが解決するのはプライバシーの問題だけではない。

- アルゴリズムが公開されることで、サービスやモデルに不正やバイアスがない証拠を示すことができる(バイアス問題の解決)

“

參考資料



<秘密分散ベース MPC vs TEE>

代表的な秘密分散ベースMPC vs TEEを紹介する

1. 準同型暗号(HE/Homomorphic Encryption)
2. マルチパーティ計算(sMPC／MPC)
3. ゼロ知識証明(ZKP)
4. カードベース暗号



<秘密分散ベース MPC vs TEE>

代表的な秘密分散ベースMPC vs TEEを紹介する

1. **準同型暗号(HE/Homomorphic Encryption)**
 - a. 完全準同型暗号(FHE/Fully HE)
 - b. 乗法準同型暗号(Multiplicative HE)
 - c. 加法準同型暗号(Additive HE)
2. マルチパーティ計算(sMPC／MPC)
3. ゼロ知識証明(ZKP)
4. カードベース暗号

● <準同型暗号(HE/Homomorphic Encryption)>

準同型暗号とは

暗号文のまま計算を行える暗号技術のこと

$\alpha(x, y)$

= Dec(Enc($\alpha(x, y)$))

= Dec($\beta(\text{Enc}(x), \text{Enc}(y))$) ←これ

x, y : データ(平文)

Enc: 暗号化計算

Dec: 復号計算

α : 目的の計算(平文用)

β : 目的の計算(暗号文用)



<準同型暗号(HE/Homomorphic Encryption)>

$\text{Dec}(\beta(\text{Enc}(x), \text{Enc}(y)))$ 公開鍵方式の準同型暗号イメージ



平文 x, y



クライアントサイド
あるいはアクセス権の絞られた環境



サーバサイド
あるいは計算担当マシン

● <準同型暗号(HE/Homomorphic Encryption)>
Dec (β (Enc (x), Enc (y))) 公開鍵方式の準同型暗号イメージ

公開鍵あるいは
クラウドキー



平文 x, y



クライアントサイド
あるいはアクセス権の絞られた環境



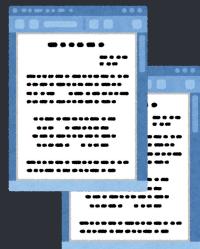
サーバサイド
あるいは計算担当マシン



<準同型暗号(HE/Homomorphic Encryption)>

Dec (β (Enc (x), Enc (y))) 公開鍵方式の準同型暗号イメージ

公開鍵あるいは
クラウドキー



平文 x, y



暗号文
Enc(x)とEnc(y)



クライアントサイド
あるいはアクセス権の絞られた環境



サーバサイド
あるいは計算担当マシン

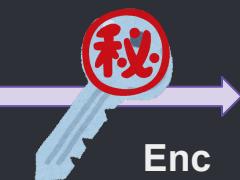
● <準同型暗号(HE/Homomorphic Encryption)>

Dec (β (Enc (x), Enc (y))) 公開鍵方式の準同型暗号イメージ

公開鍵あるいは
クラウドキー



平文 x, y



Enc



暗号文
Enc(x)とEnc(y)



クライアントサイド
あるいはアクセス権の絞られた環境



サーバサイド
あるいは計算担当マシン

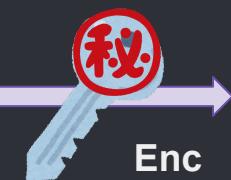
● <準同型暗号(HE/Homomorphic Encryption)>

Dec (β (Enc (x), Enc (y))) 公開鍵方式の準同型暗号イメージ

公開鍵あるいは
クラウドキー



平文 x, y



Enc



暗号文
Enc(x)とEnc(y)



クライアントサイド
あるいはアクセス権の絞られた環境



暗号文'
 β (Enc(x), Enc(y))



暗号文
Enc(x)とEnc(y)

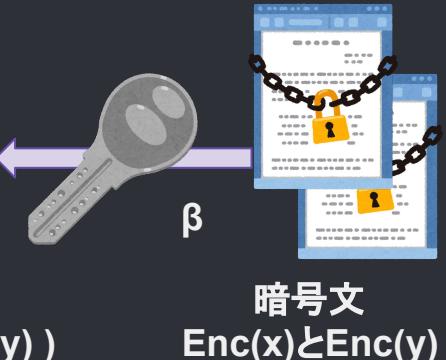
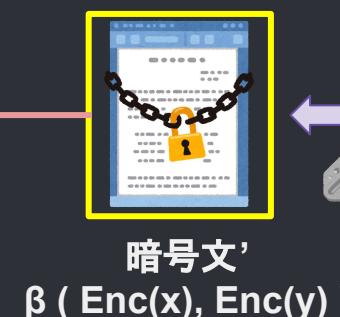


サーバサイド
あるいは計算担当マシン



<準同型暗号(HE/Homomorphic Encryption)>

Dec (β (Enc (x), Enc (y))) 公開鍵方式の準同型暗号イメージ



クライアントサイド
あるいはアクセス権の絞られた環境



サーバサイド
あるいは計算担当マシン

- <準同型暗号(HE/Homomorphic Encryption)>
 $\text{Dec}(\beta(\text{Enc}(x), \text{Enc}(y)))$ 公開鍵方式の準同型暗号イメージ

計算する側のマシンが平文を見ることはない



計算結果
の平文



Dec



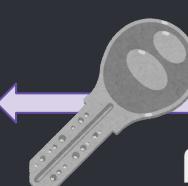
暗号文'
 $\beta(\text{Enc}(x), \text{Enc}(y))$



クライアントサイド
あるいはアクセス権の絞られた環境



暗号文'
 $\beta(\text{Enc}(x), \text{Enc}(y))$



β



暗号文
 $\text{Enc}(x)$ と $\text{Enc}(y)$



サーバサイド
あるいは計算担当マシン



<準同型暗号(HE/Homomorphic Encryption)>

Dec (β (Enc (x), Enc (y))) 公開鍵方式の準同型暗号イメージ

$\alpha (x, y)$



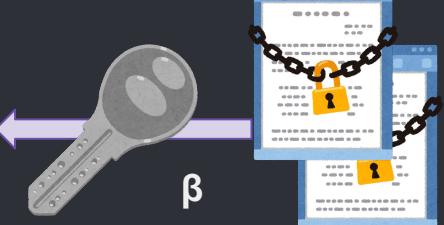
計算結果
の平文



暗号文
 $\beta (\text{Enc}(x), \text{Enc}(y))$



暗号文'
 $\beta (\text{Enc}(x), \text{Enc}(y))$



暗号文
 $\text{Enc}(x)$ と $\text{Enc}(y)$



クライアントサイド
あるいはアクセス権の絞られた環境



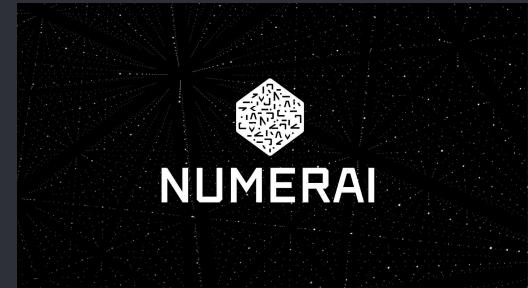
サーバサイド
あるいは計算担当マシン

● <準同型暗号(HE/Homomorphic Encryption)>

準同型暗号の種類

- 完全準同型暗号／乗法準同型暗号／加法準同型暗号
 - 以下のような性質でバリエーション
 - パフォーマンス(データサイズ、計算速度)
 - Refresh計算の要否、またはRefreshなしに可能な計算回数
 - 秘密鍵:秘密=1:多のもの、秘密鍵:秘密=多:1のもの、秘密鍵:秘密=多:多のもの
- 【番外】暗号化後も暗号化前と同じ大小関係が保たれるもの
 - 順序保存暗号(OPE/Order Preserving Encryption)
 - 選択平文攻撃が弱点である性質から、等価式あるいは大小比較で入力値が容易に割れてしまうため、準同型暗号には数えられない

● <準同型暗号(HE/Homomorphic Encryption)>
代表的なユースケース Numerai



- AIで運用が行われるヘッジファンド
 - Kaggleのように不特定多数のデータサイエンティストが学習モデルを構築し、一定の評価でランキング付け
 - 上位のAIに実際の運用を行わせる。運用実績のランキング上位者に報酬を与える
- 一定期間ごとに配布される学習データはすべて暗号化されている
 - データのフィールドも秘匿されているところがポイント
 - モデルの動作検証は別途配布されるテストデータで実施
 - 管理情報以外、属性も隠蔽されている
 - データの提供形式が少しずつ変化している

id	era	data_type	feature_intelligence1	feature_intelligence2	feature_intelligence3	feature_intelligence4	feature_intelligence5	feature_intelligence6	feature_intelligence7	feature_intelligence8
n000315175b67977	era1	train	0	0.5	0.25	0	0.5	0.25	0.25	0.25
n0014af834a96cdd	era1	train	0	0	0	0.25	0.5	0	0	0.25
n001c93979ac41d4	era1	train	0.25	0.5	0.25	0.25	1	0.75	0.75	0.25
n0034e4143f22a13	era1	train	1	0	0	0.5	0.5	0.25	0.25	0.75

● <秘密分散ベース MPC vs TEE>

代表的な秘密分散ベースMPC vs TEEを紹介する

1. 準同型暗号(HE/Homomorphic Encryption)
2. マルチパーティ計算(sMPC／MPC)
 - a. Circuit Garbling型
 - b. 秘密分散(Secret Sharing)型
 - i. SPDZ
 - ii. MASCOT
 - iii. BDOZ
 - iv. TinyOT
 - v. MiniMAC
 - vi. その他続々

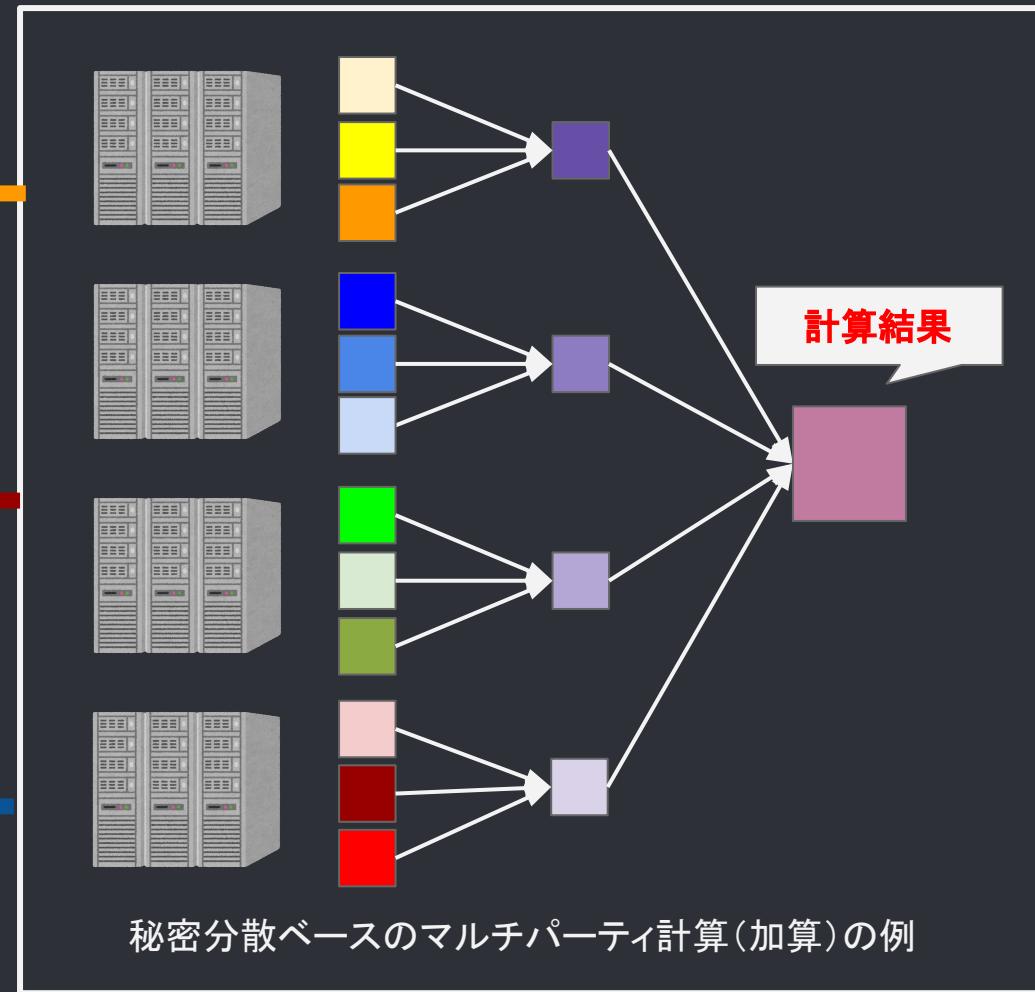
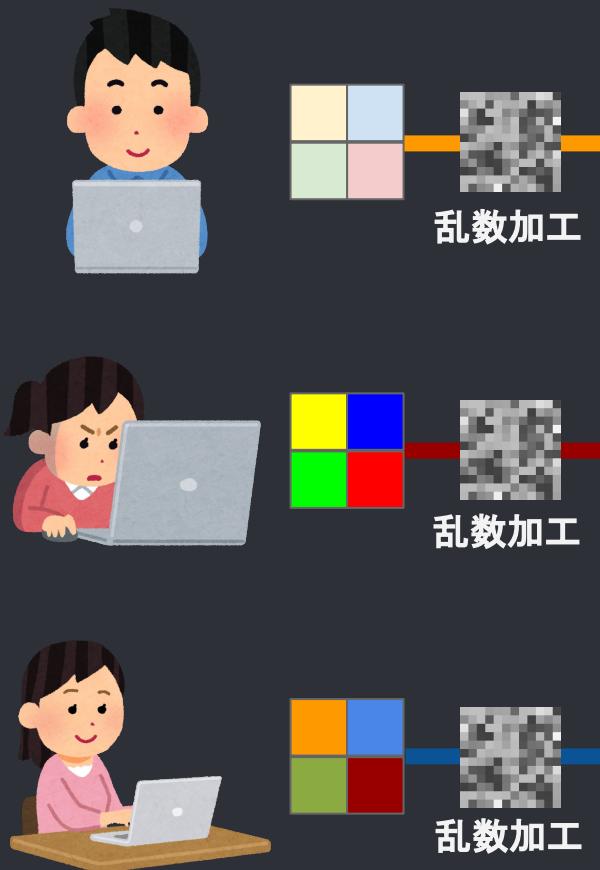
【補足】OT/Oblivious Transfer

3. ゼロ知識証明(ZKP)
4. カードベース暗号

● <マルチパーティ計算(MPC/Multi-Party Computation) >

マルチパーティ計算とは

複数のノード上で秘密を分散して保有し、ノード間で協力しあって計算結果を導く手法のこと。主にGarbled Circuitと秘密分散の2種の実装がある



● <マルチパーティ計算(MPC/Multi-Party Computation) >
秘密分散ベースのマルチパーティ計算 ~加算編~

入出力の最大値を定義し、(最大値 + 1) = 法でmodulo計算を行う

秘密A =
 $(A_0 + A_1) \% \text{法}$



$A_0 = \text{乱数A}$



$c_0 = A_0 + B_0$

秘密B =
 $(B_0 + B_1) \% \text{法}$



$B_0 = \text{乱数B}$

$r = (c_0 + c_1) \% \text{法}$

$A_1 =$
 $(\text{秘密A} - \text{乱数A}) \% \text{法}$



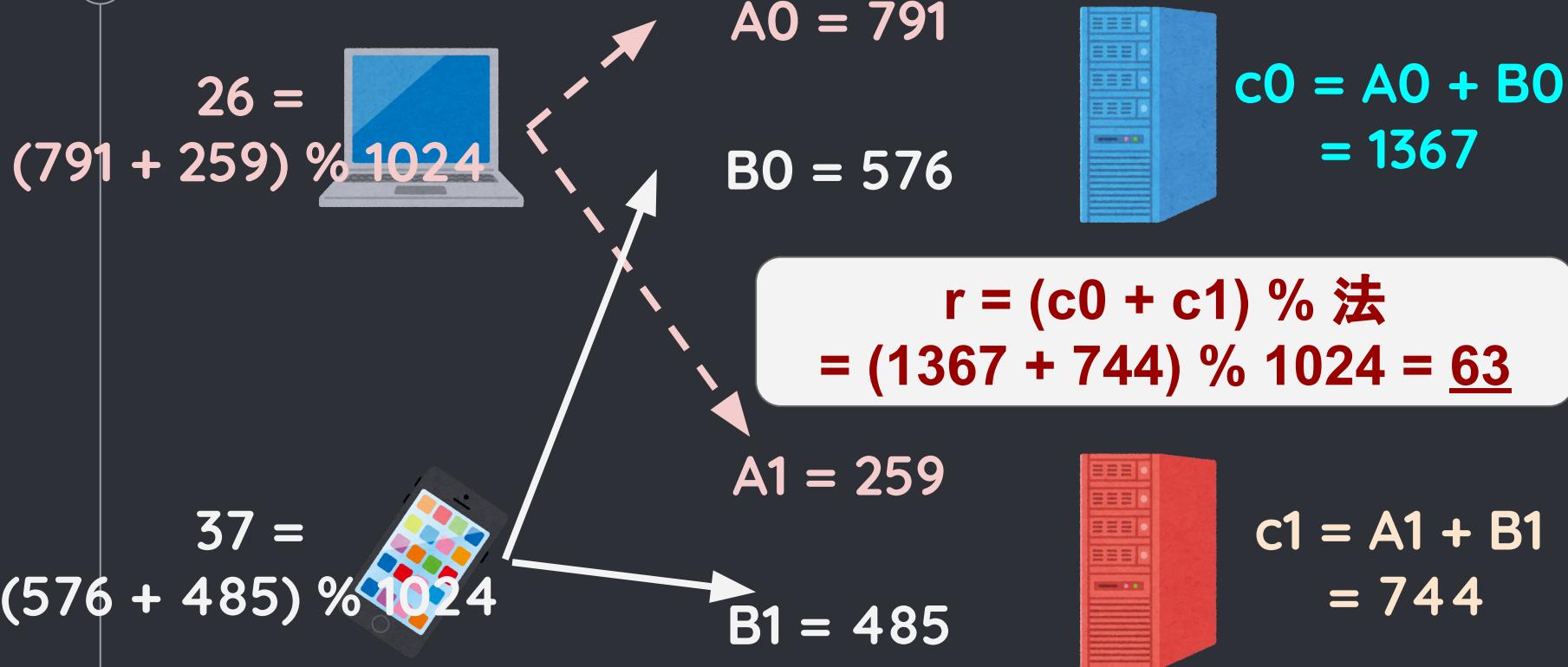
$c_1 = A_1 + B_1$

$B_1 =$
 $(\text{秘密B} - \text{乱数B}) \% \text{法}$

マルチパーティ計算のスライドでは、
『サーバ間で共有する値』は小文字で表記する



<マルチパーティ計算(MPC/Multi-Party Computation) >
秘密分散ベースのマルチパーティ計算 ~加算編~





<マルチパーティ計算(MPC/Multi-Party Computation) >

秘密分散ベースのマルチパーティ計算 ~XOR演算編~

秘密A = A0⊕A1



A0 = 亂数A



c0 = A0⊕B0

秘密B = B0⊕B1



B0 = 亂数B



c1 = A1⊕B1

A1 = 亂数A⊕秘密A

B1 = 亂数B⊕秘密B

r = c0⊕c1



<マルチパーティ計算(MPC/Multi-Party Computation) >

秘密分散ベースのマルチパーティ計算 ~XOR演算編~

$$26 = 791 \oplus 781$$



$$A_0 = 791$$



$$c_0 = A_0 \oplus B_0 \\ = 343$$

$$37 = 576 \oplus 613$$



$$B_0 = 576$$

$$A_1 = 781$$



$$B_1 = 613$$

$$r = c_0 \oplus c_1 = 360 \oplus 343 = \underline{63}$$

$$c_1 = A_1 \oplus B_1 \\ = 360$$

● <マルチパーティ計算(MPC/Multi-Party Computation) >
秘密分散ベースのマルチパーティ計算 ~積算編~

積算は加算に比べて複雑、かつ高コストである。

積算のプロセスは以下の2段階に分けることができる。

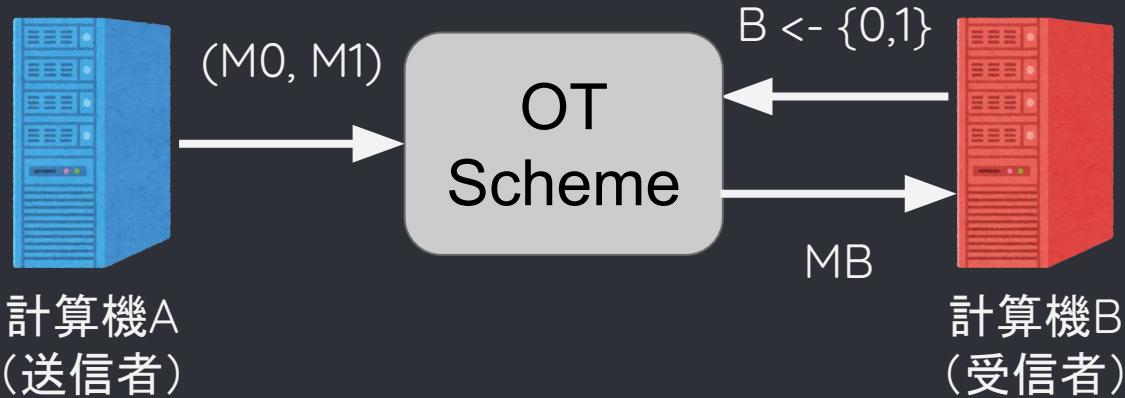
1. CR(Correlated Randomness)値の生成
 - a. 2.とは独立した処理なので、協力ノードが決まっていれば計算リクエストを受ける前から生成しておくことが可能である
2. 積算の実行
 - a. 1回の積算ごとに1.で生成したCR値を消費する(使い捨てる)
 - b. CR値を使い回すと秘密が漏れる

● <マルチパーティ計算(MPC/Multi-Party Computation) >
秘密分散ベースのマルチパーティ計算 ~積算編~

1. CR(Correlated Randomness)値の生成
 - a. 2.とは独立した処理なので、協力ノードが決まっていれば計算リクエストを受ける前から生成しておくことが可能である
2. 積算の実行
 - a. 1回の積算ごとに1.で生成したCR値を消費する(使い捨てる)
 - b. CR値を使い回すと秘密が漏れる

● <秘密計算の種類 その2—マルチパーティ計算(MPC/Multi-Party Computation)>
【補足】OT(Oblivious Transfer)の概要

例) 1 out of 2 Oblivious Transferの図



送信者は n 個のメッセージを送り、受信者はそのうちの K 個を取得する。ただし送信者はどの K 個が届いたかを知ることなく、受信者は残りの $(N-K)$ 個のメッセージを知り得ない。

● <秘密計算の種類 その2—マルチパーティ計算(MPC/Multi-Party Computation)>

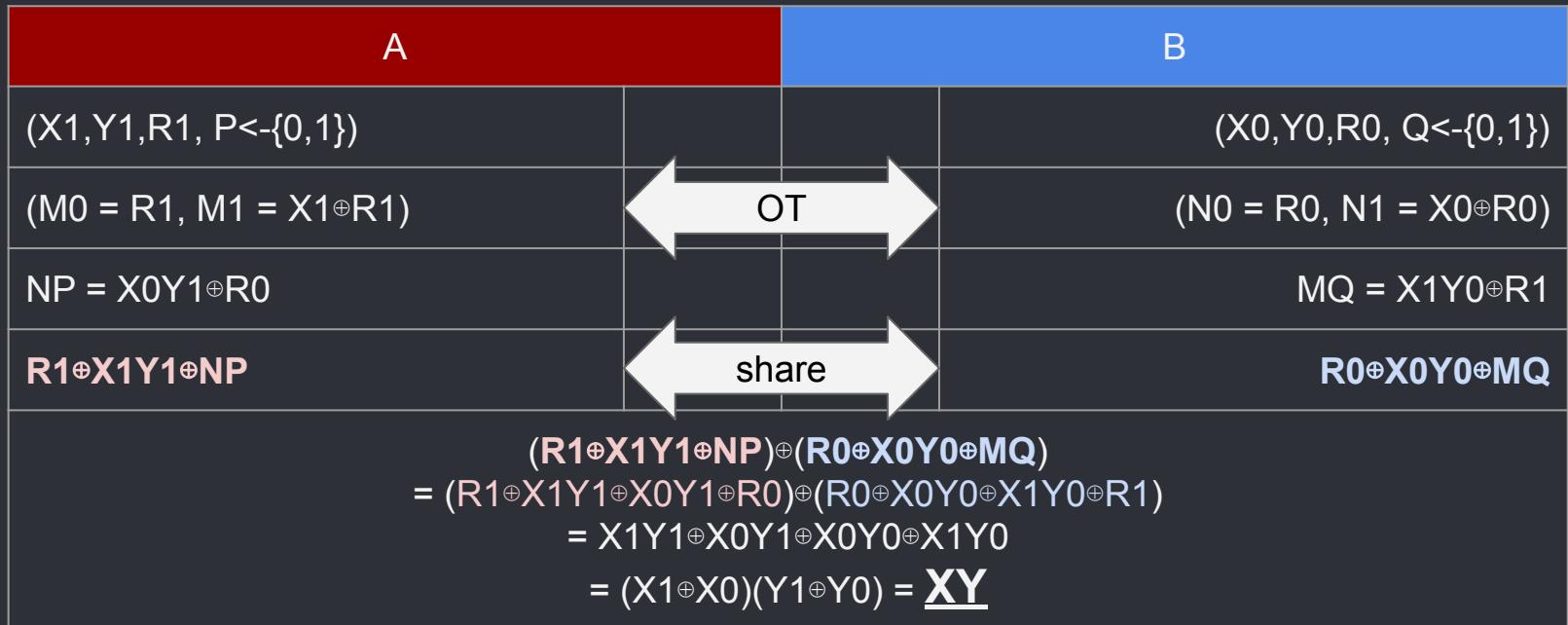
【補足】OTにより1 of 2メッセージの共有をする

Oblivious Transferのようなノード間のコミュニケーションがパフォーマンス問題の原因のひとつである。

A		B			
Action	Secret	Public	Public	Secret	Action
秘密を持つ	(M0, M1)				
鍵ペア生成	D	(n, e)		(n, e)	鍵ペア受取
乱数共有		(r0, r1)	1	(r0, r1)	乱数受取
				(B ∈ {0, 1}, K)	選択・乱数生成
選択値受取		v	2	v=(rB+K^E)%n	選択の暗号化
暗号化1	K0=(v-r0)^D%n K1=(v-r1)^D%n				
暗号化2		(n0=M0+K0, n1=M1+K1)	3	(n0, n1)	暗号文受取
Bobには $M(1-B)$ を復号できないところがポイント				MB=nB-K	選択・復号

● <秘密計算の種類 その2—マルチパーティ計算(MPC/Multi-Party Computation)>
OTを利用したCR値の生成

青サーバ(B)のシェアを (X_0, Y_0, R_0) 、
赤サーバ(A)のシェアを (X_1, Y_1, R_1) とする。
CR値として XY を導く手続きは以下の通り。



協力ノードの組み合わせが決定している場合、OTを使わず、クライアントや第三者ノードにCR値を生成してもらうアプローチがあり、その方が通信回数を減らせて効率が良い

● <秘密計算の種類 その2—マルチパーティ計算(MPC/Multi-Party Computation)>
OTを利用したCR値の生成

青サーバのシェアを($X_0=1, Y_0=1, R_0=1$)、
赤サーバのシェアを($X_1=0, Y_1=1, R_1=1$)とする。

A		B
$(X_1=0, Y_1=1, R_1=1, P<\{-0, 1\}=1)$		$(X_0=1, Y_0=1, R_0=1, Q<\{-0, 1\}=0)$
$(M_0=R_1=1, M_1=(X_1 \oplus R_1)=1)$		$(N_0=R_0=0, N_1=(X_0 \oplus R_0)=0)$
$NP=N_1=0 (=X_0Y_1 \oplus R_0=1 \wedge 1 \oplus 1=0)$		$MQ=M_0=1 (=X_1Y_0 \oplus R_1=1 \wedge 0 \oplus 1=1)$
$R_1 \oplus X_1Y_1 \oplus N_1 = 1 \oplus 0 \wedge 1 \oplus 0 = 1$		$R_0 \oplus X_0Y_0 \oplus M_0 = 1 \oplus 1 \wedge 1 \oplus 1 = 1$
$(R_1 \oplus X_1Y_1 \oplus N_1) \oplus (R_0 \oplus X_0Y_0 \oplus M_0) = 1 \oplus 1 = 0$		

● <マルチパーティ計算(MPC/Multi-Party Computation) >
秘密分散ベースのマルチパーティ計算 ～積算編～

1. CR(Correlated Randomness) 値の生成
 - a. 2.とは独立した処理なので、協力ノードが決まつていれば計算リクエストを受ける前から生成しておくことが可能である
2. 積算の実行
 - a. 1回の積算ごとに1.で生成したCR値を消費する(使い捨てる)
 - b. CR値を使い回すと秘密が漏れる

● <マルチパーティ計算(MPC/Multi-Party Computation)>
積算の実行

Multiplication TriplesあるいはBeaver's Triplesと呼ぶ

各サーバにCR値(X, Y, XY)のシェアが配布済みであることが前提。

$(A + X)(B + Y) - X(B+Y) - Y(A+X) + XY = AB$ にあたる計算をしている。

※ $d = A + X, e = B + Y$

秘密A =
 $(A_0 + A_1) \% \text{法}$



$A_0 = \text{乱数A}$



$B_0 = \text{乱数B}$

秘密B =
 $(B_0 + B_1) \% \text{法}$



$A_1 = (\text{秘密A} - \text{乱数A}) \% \text{法}$



$B_1 = (\text{秘密B} - \text{乱数B}) \% \text{法}$

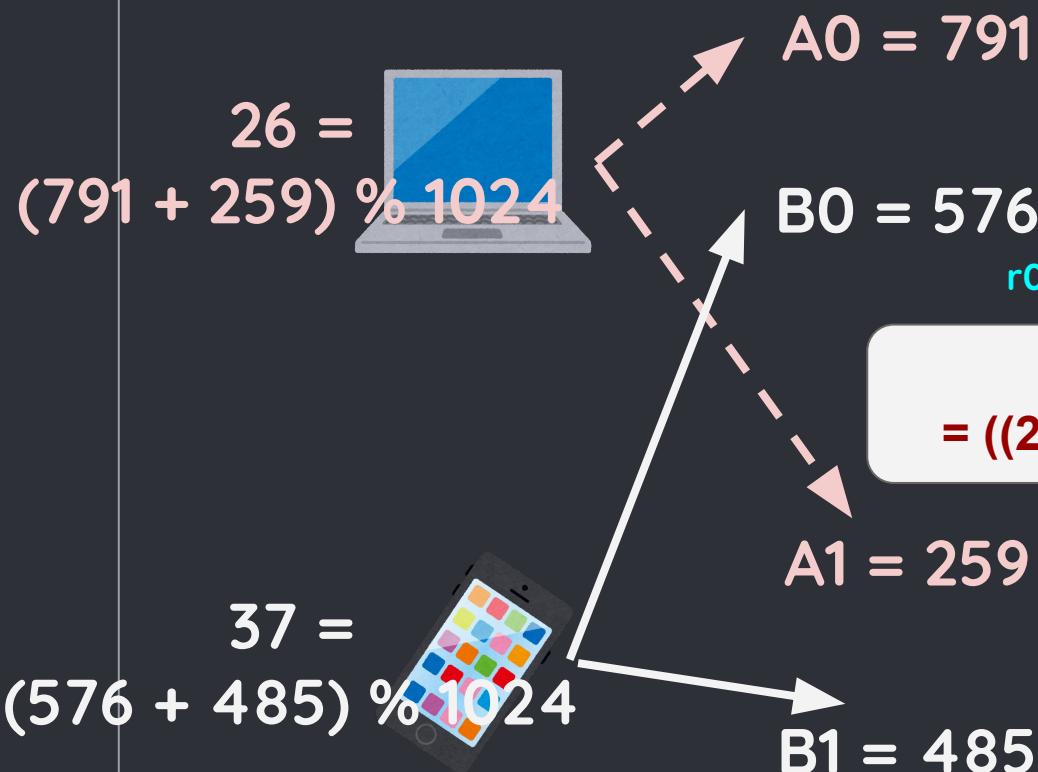
$$\begin{aligned} d_0 &= A_0 + X_0 \\ e_0 &= B_0 + Y_0 \\ d &= d_0 + d_1 \\ e &= e_0 + e_1 \\ DY_0 &= d * B_0 \\ EX_0 &= e * A_0 \\ r_0 &= DY_0 + EX_0 - XY_0 \end{aligned}$$

$r = ((d * e) - r_0 - r_1) \% \text{法}$

$$\begin{aligned} d_1 &= A_1 + X_1 \\ e_1 &= B_1 + Y_1 \\ d &= d_0 + d_1 \\ e &= e_0 + e_1 \\ DY_1 &= d * B_1 \\ EX_1 &= e * A_1 \\ r_1 &= DY_1 + EX_1 - XY_1 \end{aligned}$$

● <マルチパーティ計算(MPC/Multi-Party Computation)>
積算の実行

法を1024(以下modulo表記を省略)、CRを(X=217, Y=161, XY=34937=121)、青サーバのシェアを(X0=819, Y0=471, XY0=18211=803)、赤サーバのシェアを(X1=422, Y1=714, XY1=16726=342)とする。



Blue Server Calculations:

$$d0 = A0 + X0 = 791 + 819 = 1610 = 586$$

$$e0 = B0 + Y0 = 576 + 471 = 1047 = 23$$

$$d = d0 + d1 = 586 + 681 = 1267 = 243$$

$$e = e0 + e1 = 23 + 175 = 198$$

$$DY0 = d * Y0 = 243 * 471 = 114453 = 789$$

$$EX0 = e * X0 = 198 * 819 = 162162 = 370$$

$$r0 = DY0 + EX0 - XY0 = 789 + 370 - 803 = 356$$

$r = ((d * e) - r0 - r1) \% \text{法}$
 $= ((243 * 198) - 356 - 716) \% 1024 = \underline{\underline{962}}$

Red Server Calculations:

$$d1 = A1 + X1 = 259 + 422 = 681$$

$$e1 = B1 + Y1 = 485 + 714 = 1199 = 175$$

$$d = d0 + d1 = 243$$

$$e = e0 + e1 = 198$$

$$DY1 = d * Y1 = 243 * 714 = 173502 = 446$$

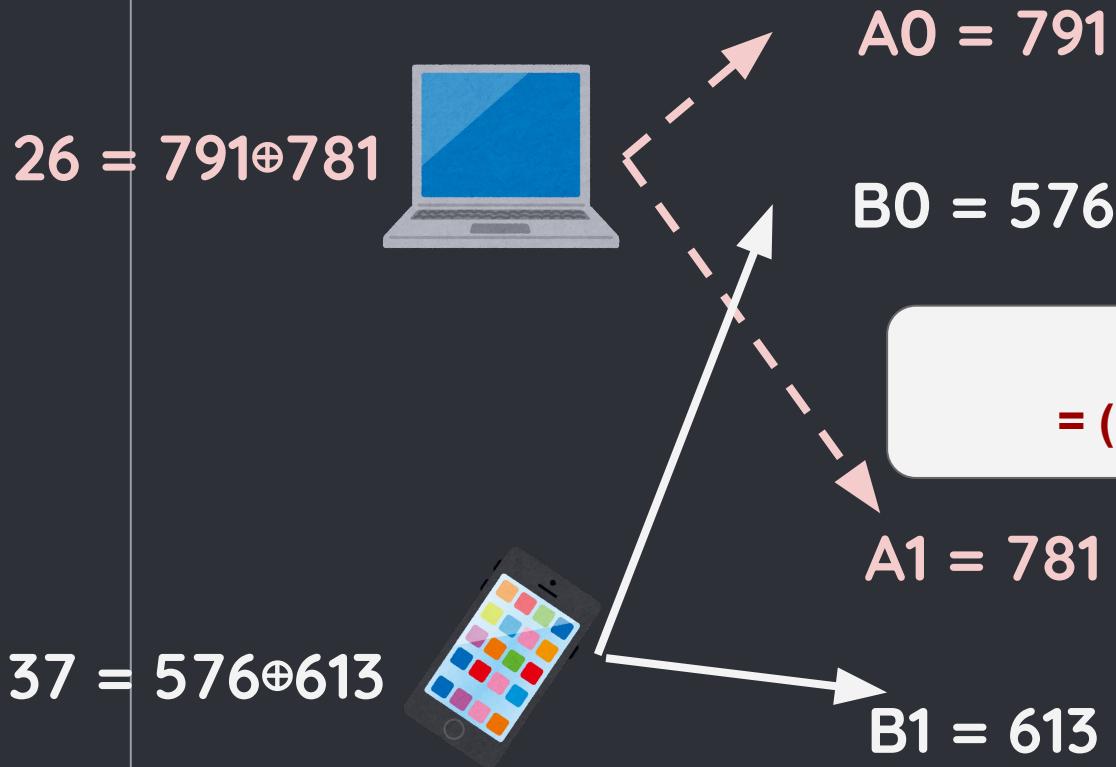
$$EX1 = e * X1 = 198 * 422 = 83556 = 612$$

$$r1 = DY1 + EX1 - XY1 = 446 + 612 - 342 = 716$$

● <マルチパーティ計算(MPC/Multi-Party Computation)>
AND演算の実行

CRを(X=217, Y=161, XY=129)、青サーバのシェアを(X0=819, Y0=471, XY0=289)、赤サーバのシェアを(X1=1002, Y1=374, XY1=416)とする。

$(A \oplus X)(B \oplus Y) \oplus X(B \oplus Y) \oplus Y(A \oplus X) \oplus XY = AB$ にあたる計算をしている。



$$\begin{aligned}
 d_0 &= A_0 \oplus X_0 = 791 \oplus 819 = 36 \\
 e_0 &= B_0 \oplus Y_0 = 576 \oplus 471 = 919 \\
 d &= d_0 \oplus d_1 = 36 \oplus 231 = 195 \\
 e &= e_0 \oplus e_1 = 919 \oplus 787 = 132 \\
 D Y_0 &= d \wedge Y_0 = 195 \wedge 471 = 195 \\
 E X_0 &= e \wedge X_0 = 132 \wedge 819 = 0 \\
 r_0 &= D Y_0 \oplus E X_0 \oplus X Y_0 = 195 \oplus 0 \oplus 289 = 482
 \end{aligned}$$

$$\begin{aligned}
 r &= ((d \wedge e) \oplus r_0 \oplus r_1) \\
 &= ((195 \wedge 132) \oplus 482 \oplus 354) = 0
 \end{aligned}$$

$$\begin{aligned}
 d_1 &= A_1 \oplus X_1 = 1002 \oplus 781 = 231 \\
 e_1 &= B_1 \oplus Y_1 = 613 \oplus 374 = 787 \\
 d &= d_0 \oplus d_1 = 195 \\
 e &= e_0 \oplus e_1 = 132 \\
 D Y_1 &= d \wedge Y_1 = 195 \wedge 374 = 66 \\
 E X_1 &= e \wedge X_1 = 132 \wedge 1002 = 128 \\
 r_1 &= D Y_1 \oplus E X_1 \oplus X Y_1 = 66 \oplus 128 \oplus 416 = 354
 \end{aligned}$$

● <秘密分散ベース MPC vs TEE>
代表的な秘密分散ベースMPC vs TEEを紹介する

1. 準同型暗号(HE/Homomorphic Encryption)
2. マルチパーティ計算(sMPC／MPC)
3. **ゼロ知識証明(ZKP)**
 - a. zk-SNARK
 - b. zk-STARK
 - c. Bulletproof
4. カードベース暗号

- <ゼロ知識証明(ZKP/Zero Knowledge Proof)>
ゼロ知識証明とは

編集中

ある知識を持つことを、知識それ自体を教えずに証明すること。

とりあえずEnigmaに使う予定はないので未編集、後日追記
いつかは使うとおもうけど(スループット向上の目的で)



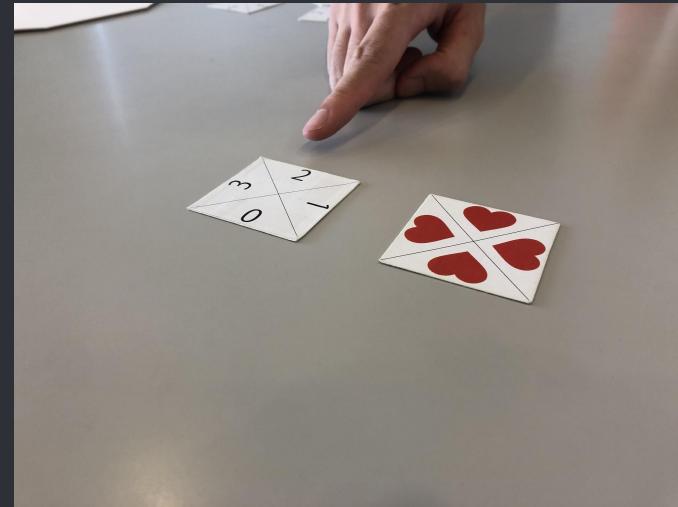
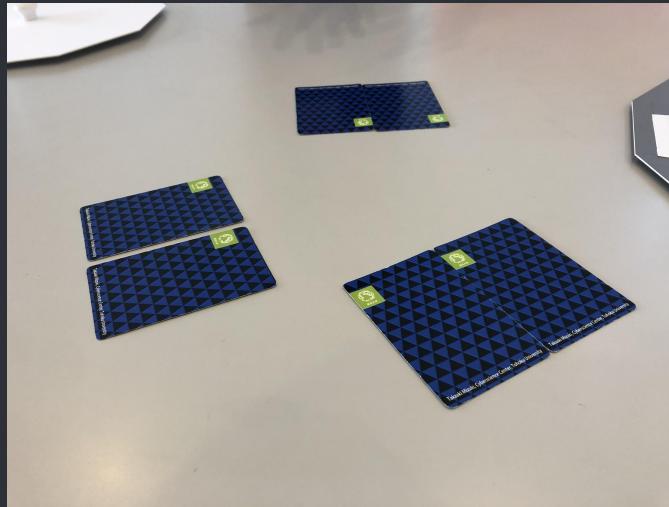
<秘密分散ベース MPC vs TEE>

代表的な秘密分散ベースMPC vs TEEを紹介する

1. 準同型暗号(HE/Homomorphic Encryption)
2. マルチパーティ計算(sMPC／MPC)
3. ゼロ知識証明(ZKP)
4. カードベース暗号

● <カードベース暗号>

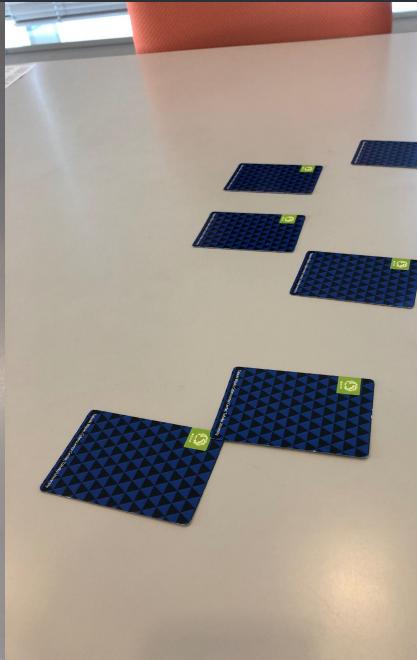
物理的なカード組などを用いて秘密計算を実現する研究分野を「カードベース暗号」と呼ぶ



※ご協力: 産総研サイバーフィジカルセキュリティ研究センター 品川様

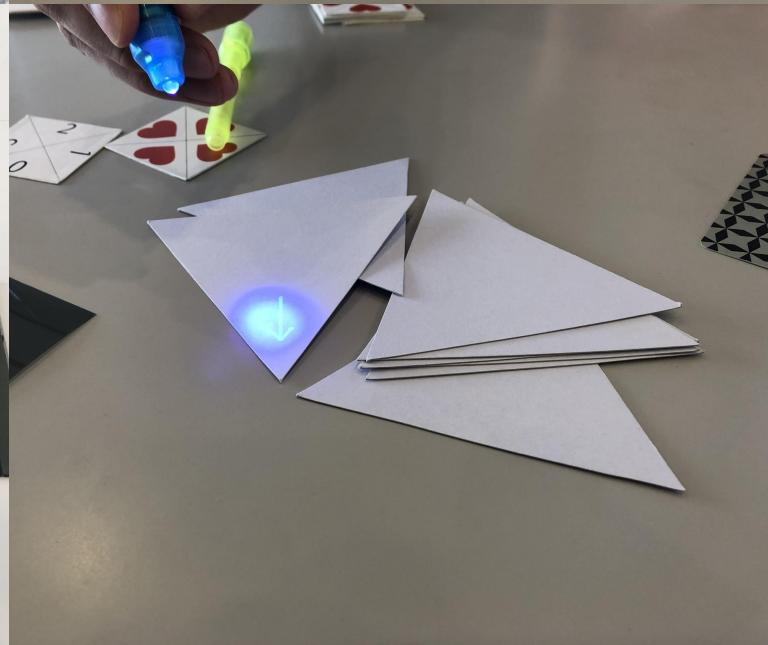
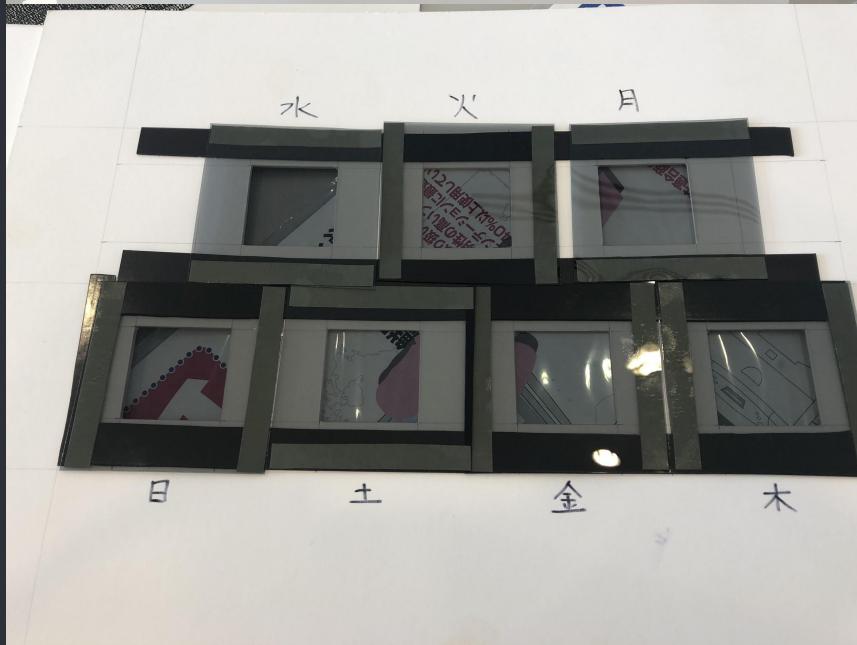
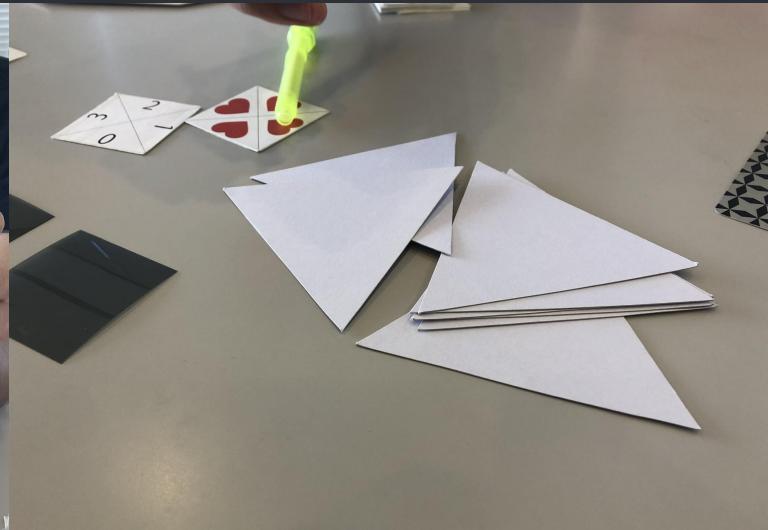
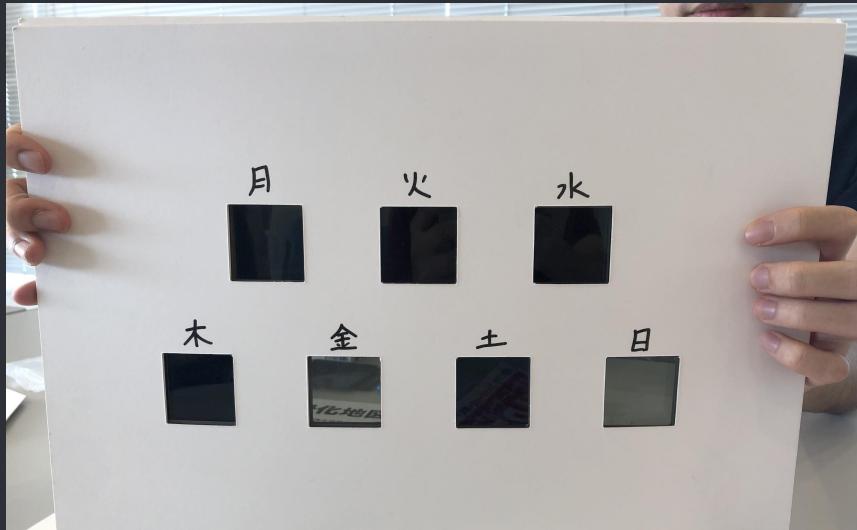
● <カードベース暗号>

3人が同じ判断を下したか(全会一致でTrueまたはFalseを選んだか)を判別できるスキーム



<カードベース暗号>

異なる形状のカード、ブラックライトインク(フリップが不要になる)、偏光板によるSet Intersection計算、OHPフィルム(動画あり)



- <シークレットコントラクト>
シークレットコントラクトが動くまで
※DISCOVERY/TEE実装の暫定版

ここからはシークレットコントラクトの作成から実行終了までの流れを説明する。

現在の実装方針に基づいた情報のため、陳腐化する箇所が多く含まれる点、ご了承いただきたい。

例)

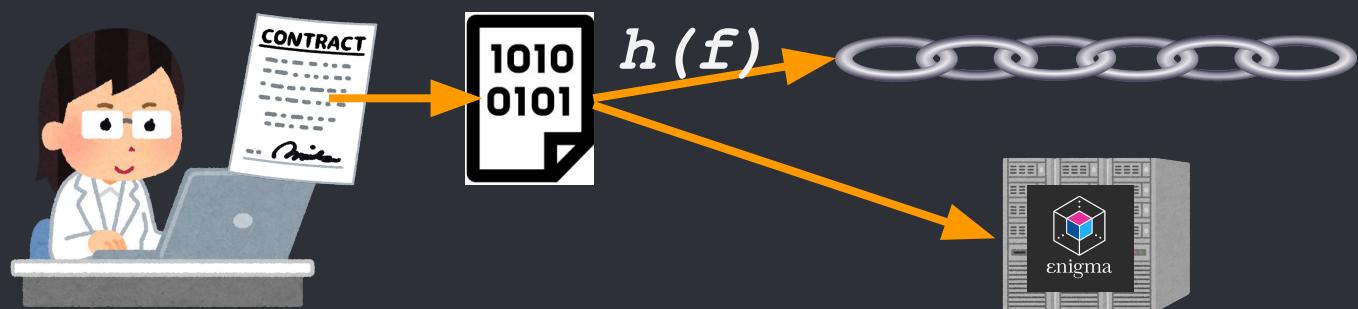
現在Enigmaの合意レイヤーはEthereumのブロックチェーンに依拠しているが、将来的にEnigmaは独自の合意レイヤーを得る予定である

● <シークレットコントラクト>

シークレットコントラクトのデプロイが終わるまで

※DISCOVERY/TEE実装の暫定版

1. シークレットコントラクトを書く ※Rust言語
 - a. 厳密にはコンパイルしてWASMのバイトコードを生成する
2. シークレットコントラクトのHash値をオンチェーンに刻む
 - a. (2019年7月現在の文脈では)EnigmaはEthereumをオンチェーンとして活用する
 - b. Enigmaは将来的に独自チェーンを持つため時間経過により文脈が変わることもある
3. オフチェーンにシークレットコントラクトをデプロイする

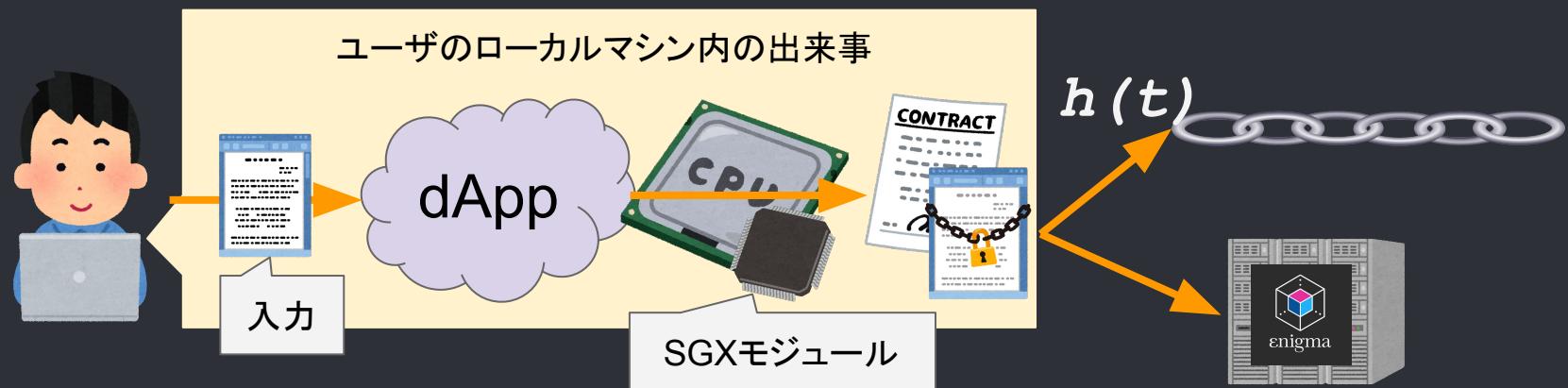


<シークレットコントラクト>

シークレットコントラクトの実行を依頼するまで

※DISCOVERY/TEE実装の暫定版

1. Task(実行対象コントラクトと入力データ他)を生成する
 - a. アプリケーション(dApp)を起動し、入力データを投入する
 - b. このとき入力データに対して、Enigma.jsライブラリによるIntel SGXをつかった命令で暗号化が施される
2. TaskのHash値をオンチェーンに刻む
 - a. このTaskのHash値を”TaskID”と呼ぶ
3. TaskをEnigmaネットワークに送る



● <シークレットコントラクト>

シークレットコントラクトの実行が終わるまで(1)

※DISCOVERY/TEE実装の暫定版

1. シークレットノード群の中から計算担当者(Worker)が選ばれる
 - a. シークレットノード = Enigmaネットワークに参加する秘密計算を担うノードのこと、いわば計算担当者の候補である
 - b. 暫定仕様ではシークレットコントラクト毎にあらかじめ計算担当者がアサインされ、一定期間(Epoch)毎に再アサインされる
 - c. Stakingが多いほど計算担当者として選ばれる可能性が高くなる。つまり計算のお仕事をもらって報酬を得られる可能性が高い

<シークレットコントラクト>

シークレットコントラクトの実行が終わるまで(2)

※DISCOVERY/TEE実装の暫定版

- - 2. 計算担当者(Worker)はEnigmaネットワーク内で計算結果を共有し、計算結果を合意する
 - 3. 所定のEthereumのスマートコントラクトを呼び出す。呼び出された Ethereumのスマートコントラクトは事後処理を担う
 - a. 事後処理はおもに残高の操作
 - b. たとえば計算担当者に対する報酬の支払いなど