金融壹账通
ONECONNECT

FiMAX

3D Zero Knowledge Protocol

Preface：

3D Zero Knowledge Protocol (3D ZKP) is designed and implemented by Ping An OneConnect's FiMAX blockchain development team. It is one of the fundamental protocols that powers FiMAX blockchain technology. Ever since its birth in 2017, FiMAX's 3D Zero Knowledge Protocol has been used in almost all of Ping An's blockchain deployments, including major headlines such as Guangdong SME financing blockchain platform, China's Custom's Tianjing Port initiative, Greater Bay Area's "Linked Port"project with China Merchant Group, and many others. It is one of the very few, if not the only, widely used zero knowledge/secure computation protocol in the world.

Authors：OCFT FiMAX Team

Frank Lu, Mu Jia, Menghan Wang, Danli Xie, Enke Liu, Pengcheng Zhang, Wei Zhang, Liming Yao, Chengyong Feng, Luyan Zha,

Max Song, Pengfei Huan, Zhuoxin Yi, Wenjing Wu, Xiang Fan, Chengkai Jia, Shiyi Fan, Fei Chen, Jun Lai, Jie Yao, Jun He, Xiaoli Zhang, Wenqiang Li, Yang Yang, Juanjuan Yu, Wei Wei, Rui Lu, Jia Guo, Xin Huang, Yanyu Chen, Fuqiang Jiang, Jingfeng Wang, Chunyan Gong, Shiwei Feng

Index

Section One: 3D Zero Knowledge Protocol Overview

1.1 Security Issues of Blockchain in Practice

1.1.1 Businesses Have Strong Demand for Data Privacy

Data has become an important factor in productivity and a key engine for competitive advantage in the business world. Unlike traditional production factors, data can be easily copied and used by competitors. Any disclosure of data for even just a very short period of time can cause economic harm on data owners. Therefore, it is not hard at all to understand why business entities across industries take data security issues more and more seriously.

As a distributed data store, blockchain disseminates data to all participants to ensure all nodes in the same blockchain are in sync. Although having a shared view of data could bring efficiency in theory, no one is willing to take the first step of sharing. Without access to data smart contracts cannot operate and promises of blockchain will be greatly limited to a few basic functions easily replaceable with traditional systems. There is little doubt that data privacy issue is the biggest roadblock for serious adoption of enterprise blockchain.

1.1.2 Blockchain's Data Dilemma

There have been many approaches in the industry to work around the data privacy issue. These approaches can be largely categorized into four categories, all carries some serious contradictions that impede adoption:

- **Confidential computing:** TEE based approach is likely not secure enough against sophisticated adversaries; Customers will be forced to deal with vendor lock-in threat since at least part of their secret keys are controlled by the vendor.
- **Hash Digest:** hashes cannot be used for any computation, making blockchain nothing but an expensive digital signature alternative; this approach may also be vulnerable to dictionary attack if not used properly.
- **Multi-ledger/channel:** Recreate data silo (p2p connections) among participants of a ledger/channel, destroying the promise of synchronized shared ledger while adding unnecessary complexity to problems that can be easily solved by traditional systems.
- **Fully Encrypted Ledger Architecture:** If data stored on blockchain are encrypted, then they are only useful to those with the secret keys to decrypt them unless very sophisticated cryptography is used (e.g. ZKP, FHE, MPC).

From both security and value proposition perspective, the fully encrypted ledger architecture offers the most promise if computational logic can be efficiently performed on encrypted data. FiMAX's 3D Zero Knowledge Protocol is designed to use advanced crypto to enable participants of blockchain data networks to cross validate its own data with other’s data, and to collectively perform advanced computations while keeping data private to their owners.

1.2 3D ZK Protocol and its Application in Real Life

3D Zero Knowledge protocol is designed to allow developers without a background in cryptography to easily codify business rules using arithmetic equations, allowing network participants to cross validate data with each other while keeping them fully private. business equations can be defined using both arithmetic operators (e.g. “+”,“-“,“*”, “/”) and comparison operators (“>”,”<”, “=”). As proof creation time and verification time per operator are both significantly less than 1 millisecond combined (more on this later), 3D ZK Protocol is both efficient and practical for large scale enterprise deployment since its birth in 2017. Applications for 3D ZK Protocol can be largely categorized into three groups explained in the next three sub-sections.

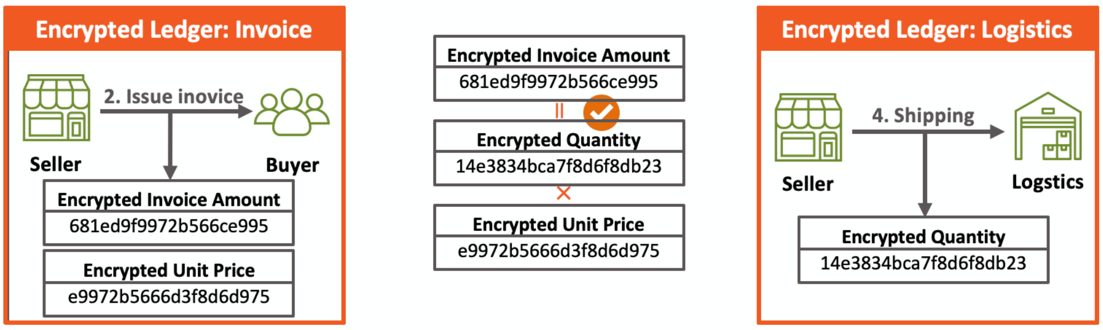
1.2.1 Zero Knowledge verification with predefined business rules

One of the most common applications of FiMAX blockchain is to allow data of various types, origins, and roles to cross validate and test their truthfulness. For example, blockchain can connect regulators and financial institutions to monitor and cross validate trade data through contracts and invoices provided by oversea exporters; logistic bills provided by logistic providers; and bill of entries, purchase orders, and importer contracts provided by domestic importers.

To achieve the business goal of the aforementioned scenario, there are two data privacy related prerequisites that must be fulfilled: First, data still belong to data owners and no one else. Second, encrypted data can be used in business logic while stay encrypted.

3D Zero Knowledge Protocol is a system that allows encrypted data to be plugged into any arithmetic equation for data verification.

Figure 1.1 Cross validation of different types of data



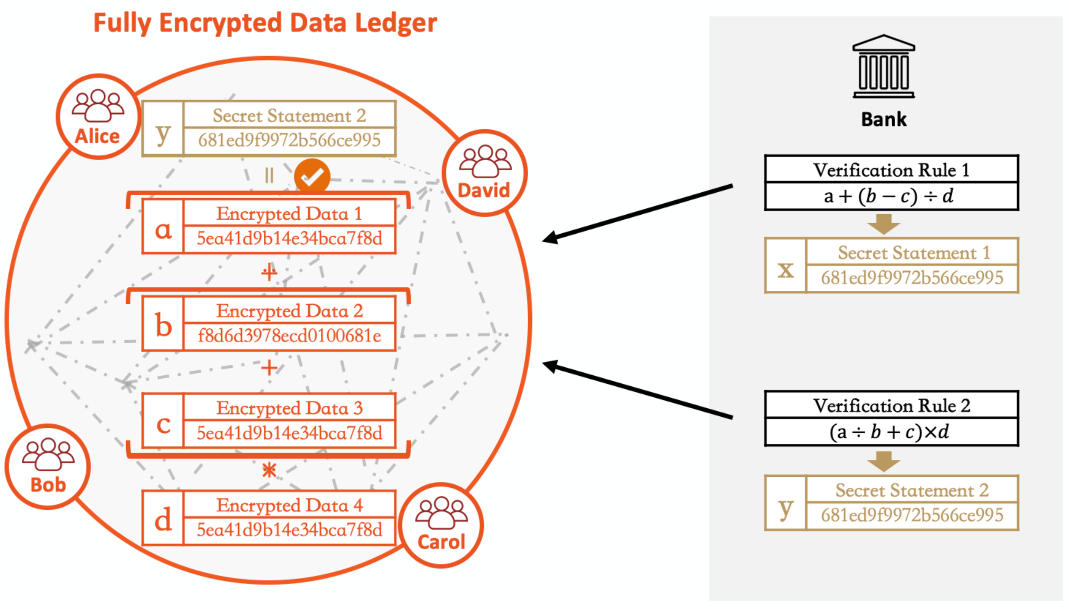
Taking the trade scenario as an example in Figure 1.1, blockchain stores encrypted invoice data from sellers, purchase order data from buyers, and logistic bill data from logistic companies. With 3D ZKP technology, participants can validate trade data without actually seeing the data it tries to validate. As figure 1.1 illustrates, invoice data from a seller are cross validated with logistic data from a shipping company (total amount = unit price * number of shipment) in zero knowledge. The verification logic is coded into smart contract. If the total amount on the invoice cannot pass the validation logic specified in the smart contract then the transaction will fail.

The potential of blockchain lays on the promise of its ability to create central record of truth. Such promise can only be fulfilled when data is available for others to use while they are still private to their owners, a dilemma can be solved by 3D Zero Knowledge Protocol.

1.2.2 Zero Knowledge verification of encrypted data with dynamically defined business rules

In the last section, we explained that we can code ZK verification logic into smart contracts to verify data to be submitted to the blockchain. But it is not reasonable to assume that all related data for a business transaction will be stored on a blockchain. Furthermore, while there are some pre-defined business validation rules can be coded into smart contracts, many more need to be defined dynamically based on business rules not known when the network was built.

Figure 1.2 Dynamic Processing of Arithmetic Equations


Fully Encrypted Data Ledger
Alice
David
Bob
Carol
y Secret Statement 2 681ed9f9972b566ce995
=
a Encrypted Data 1 5ea41d9b14e34bca7f8d
+
b Encrypted Data 2 f8d6d3978ecd0100681e
+
c Encrypted Data 3 5ea41d9b14e34bca7f8d
*
d Encrypted Data 4 5ea41d9b14e34bca7f8d
Bank
Verification Rule 1
a + (b − c) ÷ d
x Secret Statement 1 681ed9f9972b566ce995
Verification Rule 2
(a ÷ b + c)×d
y Secret Statement 2 681ed9f9972b566ce995

In Figure 1.2, Alice, Bob, Carol, and David are participants of a blockchain data network, data “a”, “b”, “c”, and “d” are encrypted data stored on that blockchain. 3D ZK Protocol allows third party entities in and outside of the blockchain network to cross validate its data with encrypted data stored on that blockchain. When a fifth participant Evan wants to check the validity of data“y”, it defines an arithmetic equation (e.g. $a + (b - c) \div d$), with input parameters being the references to encrypted data on blockchain, to describe the validation rule and sends it to the data owner. Upon receiving the request, data owner will process the arithmetic equation and generate the proof for the validation accordingly and send the proof back to Evan.

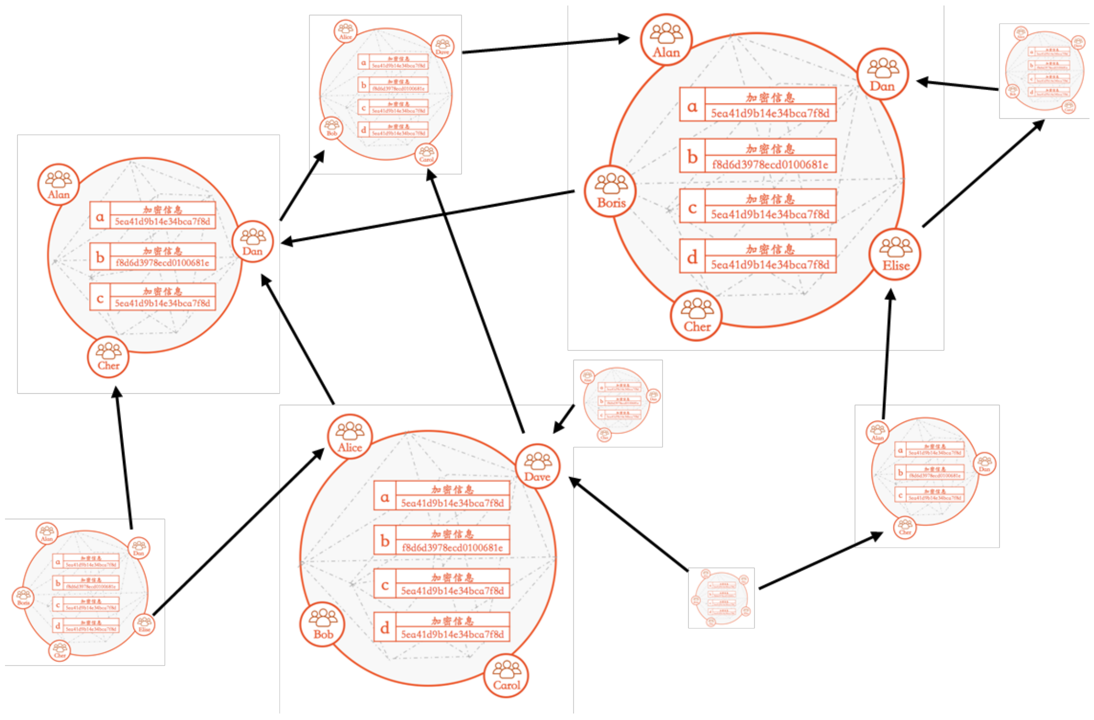
As we can see from this example, the encrypted data on blockchain was never shared to Evan, but Evan can still validate the truthfulness of his data by leveraging the data on blockchain using the business validation logic it defined in real time.

1.2.3 Connecting Blockchain Data Silos

Almost every large enterprise nowadays run a few experimental blockchain projects. Unfortunately, most of these blockchain deployments become their own disconnected data silos overtime, completely departs from the promise the same technology advertised in the beginning. It is easier said than done to connect them due to data privacy issues. Consider many blockchain platforms use sub-chains (or sub-ledger, channels) to offer data protection, interconnecting blockchains would immediately expose clear text data to connecters.

3D ZKP promotes a great ambition to interconnect blockchain networks because data are guaranteed to be safe between networks. Participants of one network can create query on data of another blockchain. As we will explain in later sections, 3D ZKP and FiMAX's fully encrypted ledger architecture not only offer a way to cross validate data, they also pave the way for data collaboration across business entities and can serve as the building block for other data collaboration technologies such as federated machine learning (more on this later).

Figure 1.3 Interconnecting Blockchain Data Silos with 3D ZKP



1.3 Applications of 3D Zero Knowledge Protocol

1.3.1 China Custom’s Tianjing Port Blockchain Initiative

Launched in April 2019, Cross-border trade transactions not only comprise buyers, sellers, and logistics but also involve indirect stake holders from different regions and jurisdictions. Transaction data could include sensitive business data and subject to stringent regulatory requirements. These restrictions will make digitization of data difficult, and workflows cannot be automated if data cannot be accessed digitally.

Businesses also have no incentive to share data with partners and financial institutions, making it hard for regulators to monitor trade transactions and difficult for financial institutions to assess financial risk.

With FiMAX blockchain and its 3D ZKP technology, data of different origins can now be cross validated without privacy concern. Regulators can leverage blockchain data to monitor trading activities in real time, and financial institution can better asses the financial risk of its customers and therefore be more confident in making business loans.

1.3.2 Guangdong SME Financing Platform

Launched in January 2020. Guangdong SME Financing platform borrows our experience from HKMA’s eTradeConnect. It aims to provide a platform that connects banks and SMEs needing business loans. Since data related to business loan transactions are sensitive to banks, all data on blockchain are encrypted and every data field is encrypted with a unique secret key.

When making a business loan request, 3D Zero Knowledge Protocol can cross validate data from sellers, buyers, and logistic providers to ensure they are legitimate and detect potential frauds early in the loaning process. Banks can cross check with other banks connected to the blockchain network to detect over/double financing activities from bad actors.

As of first quarter 2020, the Guangdong SME financing platform has already connected 213 types of data from 26 government sectors, and has collected data for 11 million SMEs based in Guangdong province. 129 financial institutions either connected to or registered on the blockchain based financing platform including China Construction Bank, China Commerce Bank, and Ping An Bank.

1.3.3 Blockchain Cross-border Supply-chain Monitoring System

Launched in the third quarter 2020 and owned by a large regional inter-government collaboration organization. This system records trade data from importer/exporters, local/foreign customs, outbound/inbound logistics, and buyer/seller agents from various countries, allowing regulators to have a complete view on cross border trades. Unlike traditional blockchain solutions where traceability is offered by either linking clear text data or incomputable hash digests, FiMAX's 3D ZKP solution goes one step beyond and links encrypted data.

In addition, the platform can automatically match and link information from different participants for the same trade. For example, product category, quantity, production date and other information are automatically matched and cross-verified in the cipher state, preventing the occurrence of false labeling of product origin and other types of misinformation.

Section Two: API Examples of 3D ZK Protocol

2.1 3D ZKP Operation Modes

3D ZKP offers two types of validation mode: the smart contract validation mode where smart contract performs zero knowledge validation of transaction inputs, and the dynamic validation mode where data are simply pulled from blockchain and validated by off-chain applications.

- **Smart Contract Validation Mode**：Validation logics are pre-defined and coded into smart contracts. Blockchain smart contracts run their validation logic on every transaction passes them. Data on blockchain can only be altered if the transaction passes the smart contract validation phase.
- **Dynamic Validation Mode**：Validation logics are defined dynamically (in real time) and verified individually by validation requesters. Validation requesters may or may not be participants of a blockchain network. They use arithmetic equation to define validation logics and send them to data owners. After receiving proof transcripts from data owners, validation requesters use both encrypted data on blockchain and proof transcripts to validate the data they intent to validate.

3D ZKP APIs：

```
/** 3D ZKP smart contract Validation Mode **/
public interface ZKPEquationService {
    /** send validation request **/
    boolean submitRequest(String id, ZKPReqParams
req);
     /** query result, tx accepted if successful **/
    ZKPProofResult queryZKPProof(String id);
}

/** 3D ZKP dynamic validation mode **/
public interface RTZKPEquationService {
    /** send validation request **/
    boolean provideZKPRequest(String id, ZKPReqParams
req);
}


/** Request parameters **/
@Data
public class ZKPReqParams {
    private String equation;
    private Map<String, DataPath> dataPath;
}

/** Query Results **/
@Data
public class ZKPProofResult {
    private String id;
    private boolean isSuccess;
    private ZKPReqParams params;
}
```

2.2 Define Arithmetic Equation

Validation requesters use arithmetic equations to define business rules:

```
 (a * b + c) * d + 100 == e / f - 10
```

alphabet letters are references to data on blockchain. Equations on the left and right sides of the comparator “==”may reference data from the same blockchain or from two different blockchains. There are two ways to reference blockchain data: through direct specification of data address on blockchain, and through a SQL statement to specify its path.

Direct specification of data address: provide type, dataID, jsonPath of the data tries to reference. jsonPath is used to reference a specific Field of a data structure (more on this later)

```
@Data
public class DataPath {
    private String type;
    private String dataID;
    private String jsonPath;
}
```

SQL statement specification: provide the SQL reference to the data of interest. Note that data owner can easily run the query since it has access to decrypted data. Optionally, data owners can provide the proof transcript to prove it has queried the correct data.

Example: return type where amount is greater than 100 from jsonPath

```
SELECT jsonpath FROM type WHERE name = 100
```

2.3 FiMAX Data Field Types

One of the highlights of FiMAX blockchain is that it adopts fully encrypted ledger architecture, where every data field on FiMAX blockchain is encrypted. FiMAX developers need to specify the encryption type for each data field, and there are three encryption types to choose from: plain (not recommended), symmetric encryption (default), and Pedersen Commitment (required for secure computation such as ZKP and MPC protocols).

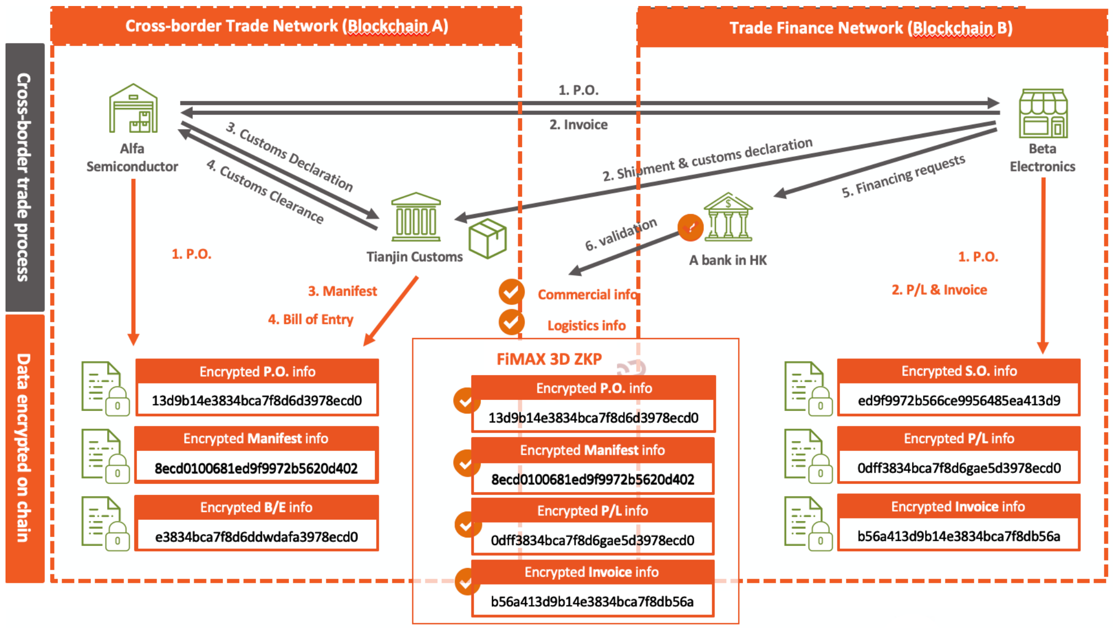
```
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface DataField {
    /** field name **/
    String alias() default "";
    /** encryption type; defaults to symmetric
encryption **/
    DataFieldEncrypt encrypt() default
DataFieldEncrypt.SYM;
    /** floating point **/
    int exponent() default 0;
    /** specify if negative **/
    boolean negative() default false;
}

public enum DataFieldEncrypt {
    /** plain text **/
    PLAIN(0),
    /** symmetric encryption **/
    SYM(1),
    /** Pedersen commitment **/
    ZKP(2);
    /** ... **/ -->
}
```

2.4 Example Application：Trade Finance Solution

2.4.1 Background

Figure 2.1 3DZKP Application in Trade Finance Workflow



Beta Electronic, an upstream buyer, purchased electronic chips from an oversea supplier Alpha Semiconductor. To complete the production of the purchase, Beta Electronics borrows money from a local bank to finance its operation.

In practice, there could be multiple blockchain networks storing data related to this trade, and participants of a trade could include government agencies (e.g. customs and tax department), global logistic providers, and many others. To make our example easier to understand, we will just demonstrate how financial institutions validate data submitted by Alpha Semiconductor and Beta Electronic. We also assume that data on blockchain have already been cross validated.

2.4.2 Transaction Validation

Beta Electronic (buyer) created a purchase order request, and Alpha Semiconductor (seller) created the invoice for the sell. Beta Electronics needs to obtain financing from a bank to complete product delivery due to capital turnover issues.

Before applying for financing, Beta Electronic submitted its invoice, customs declaration and the counterparty’s purchase order data to the blockchain. Once the submitted data passed validation performed by smart contracts, they will be written to the blockchain and made available for third party financial institutions to query and verify.

In this example, Beta Electronic made three purchase orders（*Alpha-order-001，Alpha-order-002，Alpha-order-003*）, and made an agreement with Alpha Semiconductor to use the 3-6-1 payment method (30% prepayment, 60% mid-term payment, and 10% final payment). Beta Electronic also created two invoices (Beta-invoice-001, Beta- invoice-002) for the order; the first invoice is for the first two orders, Alpha Semiconductor shipped them with the same packing number from the customs （*Customs-packing-001*）, and the second invoice is for the final order received（*Customs-packing-002*）by Beta Electronic.

Third party validation requester can now use arithmetic equation to define the relationship among purchase order, invoice, and bill of entry (from customs) as follows:

```
/*************************************************
* order-1-unit-price: order 1 unit price;
* order-2-unit-price: order 2 unit price;
* order-3-unit-price: order 3 unit price;
* packing-1-num: Custom Declaration 1: Num. of goods;
* packing-2-num: Custom Declaration 2: Num. of goods;
* invoice-1-exchange-rate: Invoice 1: exchange rate;
* invoice-1-total-amount: Invoice 1: Total Amount;
* invoice-2-exchange-rate: Invoice 2: exchange rate;
* invoice-2-total-amount: Invoice 2: Total Amount;
*************************************************/
3 * ((order-1-unit-price + order-2-unit-price) *
packing-1-num + order-3-unit-price * packing-2-num)
== 10 * (invoice-1-exchange-rate * invoice-1-total-
amount + invoice-2-exchange-rate * invoice-2-total-
amount)
```

- Beta Electronic creates the proof for the validation logic defined above by calling API `RTZKPEquationService.provideZKPRequest`, the JSON input for the request is defined here：

```
{
  "id": "Beta-invoice vs Alpha-order verification-
001",
  "req": {
    "equation": "3 * ((order-1-unit-price + order-2-
unit-price) * packing-1-num + order-3-unit-price *
packing-2-num) == 10 * (invoice-1-exchange-rate *
invoice-1-total-amount + invoice-2-exchange-rate *
invoice-2-total-amount) ",
    "dataPath": {
      "order-1-unit-price": {
        "type": "PURCAHSE_ORDER",
        "dataID": "k",
        "jsonPath": "unitPrice"
      },
      "order-2-unit-price": {
        "type": "PURCAHSE_ORDER",
        "dataID": "Alpha-order-002",
        "jsonPath": "unitPrice"
      },
      "order-2-unit-price": {
        "type": "PURCAHSE_ORDER",
        "dataID": "Alpha-order-003",
        "jsonPath": "unitPrice"
      },
      "packing-1-num": {
        "type": "PACKING",
        "dataID": "Customs-packing-001",
        "jsonPath": "goodsNum"
      },
      "packing-2-num": {
        "type": "PACKING",
        "dataID": "Customs-packing-002",
        "jsonPath": "goodsNum"
      },
      "invoice-1-exchange-rate": {
        "type": "INVOICE",
        "dataID": "Beta-invoice-001",
        "jsonPath": "exchangeRate"
      },
      "invoice-1-total-amount": {
        "type": "INVOICE",
        "dataID": "Beta-invoice-001",
        "jsonPath": "totalAmount"
      },
      "invoice-2-exchange-rate": {
        "type": "INVOICE",
        "dataID": "Beta-invoice-002",
        "jsonPath": "exchangeRate"
      },
      "invoice-2-total-amount": {
        "type": "INVOICE",
        "dataID": "Beta-invoice-002",
        "jsonPath": "totalAmount"
      }
    }
  }
}
```

- It is assumed that data on blockchain has already passed the validation, but participants on the network can still call API `ZKPEquationService.queryZKPProof` to validate the data themselves.

- Non-numerical encrypted data such as buyer name ("Beta Eletronic") can also be validated. In the following example request, validator sends the request to check if the buyer name on the encrypted invoice data match that on the purchase order. Note that buyer name field must be encrypted using Pedersen Commitment for this protocol to work.

```
{
  "id": "Beta-invoice-001 vs Alpha-order-001 buyerId",
  "req": {
    "equation": "invoice_buyer == order_buyer",
    "dataPath": {
      "invoice_buyer": {
        "type": "INVOICE",
        "dataID": "Beta-invoice-001",
        "jsonPath": "buyerId"
      },
      "order_buyer": {
        "type": "PURCHASE_ORDER",
        "dataID": "Alpha-order-001",
        "jsonPath": "buyerId"
      }
    }
  }
}
```

2.4.3 Cross Entity Data Validation

In order to further verify the authenticity of the trade and ensure the financing is indeed used for the procurement in the current contract, the bank may need to check if the information on customs declaration is consistent with that on the bill of lading.

The bill of lading is owned by Alpha Semiconductor, so the bank cannot obtain the original data from Beta Electronic. Nevertheless, the bank can again use the 3D ZK protocol to perform the validation across two blockchain networks.

- A bank sends the validation request to check if the number on customs declaration is consistent with that on the bill of lading by calling API `RTZKPEquationService.provideZKPRequest` with JSON input：

```
{
  "id": goods 001 invoice vs order",
  "req": {
    "equation": "customs_num == delivery_num",
    "dataPath": {
      "customs_num ": {
        "type": "CUSTOMS",
        "dataID": "Customs-packing-001",
        "jsonPath": "goodsNum"
      },
      "delivery_num": {
        "type": "DELIVERY",
        "dataID": "Delivery-bill-001",
        "jsonPath": "goodsNum"
      }
    }
  }
}
```

This request is sent to both Alpha Semiconductor of Blockchain A network and Beta Electronic of Blockchain B network, and validated locally by the bank itself with transcripts returned from both networks.

- The bank checks the validity of the bill of lading by making sure the date on bill of lading is no earlier than March 14, 2021

```
{
  "id": "delivery 001 expire validate",
  "req": {
    "equation": "time < 1615680000",
    "dataPath": {
      "time": {
        "type": "DELIVERY",
        "dataID": "Delivery-bill-001",
        "jsonPath": "expireDate"
      }
    }
  }
}
```

2.4.4 Example Appendix：Data Templates

Purchase Order：

```
@DataTemplate(id = "PURCHASE_ORDER")
public class DefaultPurchaseOrder {

    /** Buyer Id **/
    @DataField(encrypt = DataFieldEncrypt.ZKP)
    private String buyerId;
    /** Seller Id **/
    @DataField(encrypt = DataFieldEncrypt.ZKP)
    private String sellerId;

    /** PO unit price **/
    @DataField(encrypt = DataFieldEncrypt.ZKP,
exponent = 3)
    private BigDecimal unitPrice;

  /** ... other related fields(skip) **/`
}
```

Invoice：

```
@DataTemplate(id = "INVOICE")
public class DefaultInvoice {

    /** Buyer Id **/
    @DataField(encrypt = DataFieldEncrypt.ZKP)
    private String buyerId;

    /** Seller Id **/
    @DataField(encrypt = DataFieldEncrypt.ZKP)
    private String sellerId;

    /** Invoice identifier **/
    @DataField(encrypt = DataFieldEncrypt.ZKP)
    private String invoiceId;

    /** Invoice total amount **/
    @DataField(encrypt = DataFieldEncrypt.ZKP,
exponent = 3)
    private BigDecimal totalInvoiceAmount;

    /** Invoice exchange rate **/
    @DataField(encrypt = DataFieldEncrypt.ZKP,
exponent = 3)
    private long exchangeRate;

  /** ... Other related fields(skip) **/`
}
```

Custom declaration：

```
@DataTemplate(id = "CUSTOMS")
public class DefaultCustoms {

    /** Declaration ID **/
    @DataField(encrypt = DataFieldEncrypt.ZKP)
    private String entryId;

    /** number of goods **/
    @DataField(encrypt = DataFieldEncrypt.ZKP)
    private long goodsNum;

  /** ... other related fields(skip) **/`
}
```

Bill of lading：

```
@DataTemplate(id = "DELIVERY_BILL")
public class DefaultDeliveryBill {

     /** shipment Id **/
    @DataField(encrypt = DataFieldEncrypt.ZKP)
    private String billId;

    /** number of packages **/
    @DataField(encrypt = DataFieldEncrypt.ZKP)
    private long packNum;

    /** creation date **/
    @DataField(encrypt = DataFieldEncrypt.ZKP)
    private long createDate;

    /** expiration date **/
    @DataField(encrypt = DataFieldEncrypt.ZKP)
    private long expireDate;

  /** ... other related fields (skip) **/`
}
```

Section Three: The 3D Zero Knowledge Protocol

3D Zero Knowledge Protocol is a solution set that integrates several crypto algorithms. It is a system designed to solve complex problems by leveraging different algorithm as long as the data are encrypted using Pedersen Commitment. It enables developers without cryptography background to define business rules with arithmetic equations and run on encrypted data.

The technical concept and its implementation algorithm was first introduced in 2017, at the time when main stream ZKP and other crypto algorithms proposed for blockchain are slow, inefficient, difficult to use, and based on debatable security assumptions.

3D ZKP is the core technology behind several high profile blockchain projects including Guangdong Blockchain SME Financing Platform ,"Linked Port" with China Marchant Group, "OCEAN"platform with China Customs, and the backbone blockchain system for many of China's ministries and projects operated by international organizations.

3.1 Protocol Overview

Pedersen commitments are represented with brackets e.g. [a] , where the letter 'a'inside the brackets is the hidden data. In a Pedersen Commitment $[a] = g_1^a h_1^\alpha$, the discrete log relation between g_1 and h_1 is unknown so the commitment scheme is computationally binding. We also know for a fact that Pedersen Commitment is perfectly hiding so it is forever safe to keep the data on blockchain even if quantum computer is available today. We use Pedersen commitment as the encryption algorithm for data stored (including text) because we found it to be the lowest common denominator for many ZKP and MPC protocols. We also use the pairing group $\mathbb{G}_1$ for the Pedersen commitments because pairing function will enable more functionalities on encrypted data through integration with other crypto protocols.

Pairing function is defined by $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_t$ s.t. $\hat{e}(aP, bP) \longrightarrow \hat{e}(P, P)^{ab}$。There are primarily three types of pairing functions：Type1: $\mathbb{G}_1 = \mathbb{G}_2$；Type2: $\mathbb{G}_1 \neq \mathbb{G}_2$，but there exist an efficient function such that $\phi: \mathbb{G}_2 \longrightarrow \mathbb{G}_1$; and Type3: $\mathbb{G}_1 \neq \mathbb{G}_2$.

Pedersen commitment has additive homomorphic property, so additive operations of encrypted data on blockchain $[a] + [b]$=$[c]$ can be verified with the homomorphic property of input parameters $[a], [b]$, and $[c]$. The tricky part is to verify multiplication and division operations i.e. $\{([c], [a], [b] \in \mathbb{G}): a, b, \alpha, \beta \in \mathbb{Z}_p;\ a \cdot b = c\ \}$.

There generally two approaches to handle multiplicative operations, the most talked about approach in blockchain articles is the SNARKs approach. The most inciting part of the SNARKs approach is that it offer sub-linear verifier cost and constant proof size. However, the downside is also serious, expensive prover cost make SNARKs generally very slow and the marginal benefit of verifier efficiency only starts to appear when there are at least 1,000+ multiplicative operation. Furthermore, their reliance on constraint system and subsequent preprocessing of application circuits make SNARKs protocols unsuitable for processing dynamic created business rules explained in earlier sections.

The other approach is the trivial approach mostly used in MPC protocols. These protocols are straight forward to use and require neither constraint system nor preprocessing of application circuits. The down side of the trivial approach is their proof size is generally very high and grow linearly, and is only efficient for simple arithmetic equations.

Neither approach was suitable for us to build developer friendly APIs illustrated in section two, so we end up designing our own set of algorithms and give birth to the 3D Zero Knowledge Protocol. The concept of 3D ZKP was first introduced in 2017. We will give a detailed explanation of the original algorithms designed in 2017-2018 in this section to illustrate the idea, and then in the next section we will give a quick intro to its latest iteration - FXN.

Pairing function itself is a multiplicative operation on two group elements. However, such pairing function can only be performed once on a group element. For simplicity, we will assume Type1 pairing is used here.

We start by assuming an logical group element $i_t = \hat{e}(h, h) \in \mathbb{G}_t$. Note that we never learn the exponent x of ${g_t}^x = i_t$ but we can still get i_t using pairing function. If the two group element inputs to the pairing function are Pedersen commitments $[\mathrm{a}]$ and $[\mathrm{b}]$, then the product of the pairing is：

$$\hat{\mathrm{e}}([\mathrm{a}], [\mathrm{b}]) = \mathrm{g}_\mathrm{t}^{\mathrm{ab}} \mathrm{h}_\mathrm{t}^{\mathrm{a\beta+b\alpha}} \mathrm{i}_\mathrm{t}^{\alpha\beta} \in \mathbb{G}_t$$

Two problems: first, the product is no longer a Pedersen commitment since it now embeds a new base point i_t; Second, since the product is in $\mathbb{G}_t$ we cannot use pairing function to perform another multiplicative operation on the result again.

We use a simple trick to solve the problems mentioned above. Prover create $[c] = g^{ab}h^{\alpha\beta+b\alpha} \in \mathbb{G}$, that satisfies the relation $\{([c],[a],[b] \in \mathbb{G}): a, b, \alpha, \beta \in \mathbb{Z}_p;\ a \cdot b = c\ \}$, and then create a public key $PK_{\alpha\beta}$ such that $PK_{\alpha\beta} = i_t^{\alpha\beta}$ where the private key is $\alpha\beta$, or the multiplicative product of secret keys of $[a]$ and $[b]$. If a prover can prove the knowledge of $\alpha\beta$, then the prover can prove the hidden value of $[c]$ is the multiplicative product of hidden values of $[a]$ and $[b]$. Note that $PK_{\alpha\beta}$ is extracted by verifier using the equation below:

$$PK_{\alpha\beta} = \frac{\hat{e}([a],[b])}{\hat{e}([c],g)} = i_t^{\alpha\beta} \in \mathbb{G}_t$$

if the prover does not know the value of $\alpha\beta$, then it cannot create the proof of knowledge of $\alpha\beta$ based on the discrete log assumption. Any protocol that can prove the knowledge of $\alpha\beta$ will work. In practice, we use the digital signature algorithm defined in China's SM2 standard. The proof transcript, or the signature, of the proof is denoted by $sig_{\alpha\beta}$

Since $[c]$ is on curve $\mathbb{G}$, pairing function can be performed on $[c]$ again with another group element, allowing unlimited number of multiplicative operations to be performed on each encrypted (committed) value.

We use the same trick to prove division operation. The difference is that provers need to use the multiplicative inverses of committed values as divisors, and then provers need to provide the proofs to prove inverse commitments are valid.

For example, the modular multiplicative inverse of $\mathrm{b} \in \mathbb{Z}_\mathrm{p}$ is $\mathrm{b}^{-1} \in \mathbb{Z}_\mathrm{p}$, such that $\mathrm{b} \cdot \mathrm{b}^{-1} = 1 \in \mathbb{Z}_\mathrm{p}$. the inverse commitment of b is represented as $[\mathrm{b}^{-1}] = \mathrm{g}^{\mathrm{b}^{-1}}\mathrm{h}^{\beta'} \in \mathbb{G}$, β' is a new blinding key for $[\mathrm{b}^{-1}]$ that is not related with β, the blinding key of $[\mathrm{b}]$. the prover needs to prove that $[d] = [\mathrm{a}]/[\mathrm{b}]$ also needs to prove that $[1] = [\mathrm{b}][\mathrm{b}^{-1}]$. The full transcript is: $[\mathrm{b}^{-1}]$, $\mathrm{sig}_{\alpha\beta}$, $\mathrm{sig}_{\beta\beta'}$, $\mathrm{h}^{\mathrm{b}\beta'+\mathrm{b}^{-1}\beta}$, note that $[1] = \mathrm{g}^1 \cdot \mathrm{h}^{\mathrm{b}\beta'+\mathrm{b}^{-1}\beta}$. Verifiers can use the equations below to retract the public keys needed to verify the division operation:

$$PK_{\beta\beta'} = \hat{\mathrm{e}}([\mathrm{b}],[\mathrm{b}^{-1}])/\hat{\mathrm{e}}(\mathrm{g}^1 \cdot \mathrm{PK}_{\mathrm{b}\beta'+\mathrm{b}^{-1}\beta}, \mathrm{g}) = \mathrm{i}_\mathrm{t}^{\beta\beta'}$$

$$PK_{\alpha\beta'} = \frac{\hat{e}([a],[b^{-1}])}{\hat{e}([d],g)} = i_t^{\alpha\beta'}$$

if the prover can provide the proof of knowledge of $\beta\beta'$ and $\alpha\beta'$, then the division operation can be verified by verifiers. In summary, there are two steps in verifying division operation. first, verify $[\mathrm{b}^{-1}]$ such that $[1] = [\mathrm{b}][\mathrm{b}^{-1}]$. Second, take $[\mathrm{b}^{-1}]$ to the standard multiplication operation and verify if $[\mathrm{d}] \overset{?}{=} [\mathrm{a}][\mathrm{b}]$. To make the protocol more efficient, you can join the two public keys and proof them with just one signature.

3.2 Batch Verification of Multiplicative Operations

Verify one multiplication at a time is obviously expensive and not much better than the trivial approach. Fortunately, it is straight forward to upgrade the protocol to allow batch verification of multiplicative operations, which will considerably shrink the proof size.

The trick is to let verifier create a challenge k, which is then used as the seed to create set $\mathbf{k} = \{\mathrm{k}^1, \dots, \mathrm{k}^n\}$. The updated protocol is listed here:

$\mathcal{P}$ compute

$$[c_i] = g^{a_i b_i} h^{a_i \beta_i + b_i \alpha_i}, \ \ i = \{1, \dots, n\}$$

$\mathcal{P} \to \mathcal{V}: [c]$

$\mathcal{V} \to \mathcal{P}: k \overset{\$}{\leftarrow} \mathbb{Z}_p$

$\mathcal{P}$ compute

$$k = \{k^1, \dots, k^n\}$$

$$s = \sum_1^n \alpha_i \beta_i \cdot k^i$$

$$\mathrm{sig}_s \Leftarrow \mathrm{sign}(s, \mathrm{message}) \qquad \text{//message}$$

$\mathcal{P} \ \to \mathcal{V}: \mathrm{sig}_s$

$$\mathrm{PK}_s = \frac{\prod_1^n \hat{e}([a_i],[b_i])^{k^i}}{\hat{e}\left(\prod_1^n [c_i]^{k^i}, g\right)} \in \mathbb{G}_t$$

Verifier use sig_s to verify PK_s. So instead of sending proof of knowledge for each multiplicative operation, the whole batch can be verified using just one signature. In practice, both Prover and Verifier use Fiat-Shamir heuristic to generate challenge k to make the protocol non-interactive.

It is worth noticing that if Type2 or Type3 pairing is used, some commitments need to be on $\mathbb{G}_2$ before it can be used in pairing function，and verifiers need to verify if the value on $\mathbb{G}_2$ match that on $\mathbb{G}_1$: $\hat{e}\left(\prod_1^n([a_i]_1 \cdot l^i), g_2\right) \overset{?}{=} \hat{e}(g_1, \prod_1^n([a_i]_2 \cdot l^i))$.

Now we got a working protocol that we can use as the underlying algorithm for the APIs explained in section 2. It is not perfect in today's standard, but it was secure, efficient, and easy to implement, and probably the best option other there in 2017. We have certainly come a long way since then. before we give a short intro to the underlying algorithm FXN we use today, we will first explain the fact-hiding protocol in the next sub-section, which is an absolutely critical feature in 3D ZKP.

3.3 Fact Hiding ZK Protocol

Allowing network participants to dynamically use arithmetic equations to specify verification rules create privacy problems on its own, because information may leak when the same data records are used in multiple arithmetic equations. In addition, data to be verified also need to be sealed to prevent provers learn business secrets from verification requests.

We therefore need a protocol that can not only hide the verification fact from provers but also protect data leaked to verifiers through overlapping verification rules. We call this protocol “fact-hiding”, and it can be plugged into any ZK protocol as long as the data fact is encrypted using Pedersen Commitment.

Basic Workflow：

Bob wants to verify his data fact $[\mathrm{b}]$ from Alice’s data using the verification rule defined by arithmetic equation $E()$ such that $E() = [p_1] + \frac{[p_2]}{[p_3]} * ([p_4] - [p_5])$, where $[p_1], \dots, [p_5]$ are encrypted data stored on blockchain. $[a]$ is the data Alice calculated from the equation $E()$ using $[p_1], \dots, [p_5]$ such that $[a] = [p_1] + \frac{[p_2]}{[p_3]} * ([p_4] - [p_5])$, and Bob wants to know if $[b] \overset{?}{=} E()$ or $[b] \overset{?}{=} [a]$. Let’s assume:

$$[a] = g^a h^x$$

$$[b] = g^b h^y$$

Step One：

Bob sends the verification request to Alice, the request includes the verification equation $\mathrm{E}()$ and data to be verified $[\mathrm{b}]$.

Step Two：

Alice use her data as inputs to $\mathrm{E}()$ and to get $[\mathrm{a}]$ (note since Alice has access to her own data, so the hidden value of $[\mathrm{a}]$ is calculated in clear text and the result a is encrypted using Pedersen Commitment. Alice also generates and sends transcripts $[\mathrm{a}'], [\mathrm{b}'], \mathrm{v11}, \mathrm{v12}$ to Bob along with $[\mathrm{a}]$.

$$[\mathrm{a}'] = [\mathrm{a}]^{\alpha} = \mathrm{g}^{\mathrm{a}\alpha}\mathrm{h}^{\mathrm{x}\alpha}$$

$$[\mathrm{b}'] = [\mathrm{b}]^{\alpha} = \mathrm{g}^{\mathrm{b}\alpha}\mathrm{h}^{\mathrm{y}\alpha}$$

$$v11 = \hat{e}(h^{x\alpha}, g_2) = h_t^{x\alpha}$$

$$v12 = h^{\alpha}$$

Step Three：

Bob perform the final verification by checking whether $[b] \overset{?}{=} [a]$.

First, get the product of a and α on $\mathbb{G}_t$:

$$\hat{e}([a'], g_2) = g_t^{a\alpha} h_t^{x\alpha}$$

$$g_t^{a\alpha} h_t^{x\alpha}/v11 = g_t^{a\alpha} h_t^{x\alpha}/h_t^{x\alpha} = g_t^{a\alpha}$$

Second, get the product of b and α on $\mathbb{G}_t$:

$$\hat{e}([b'], g_2) = g_t^{b\alpha} h_t^{y\alpha}$$

$$\hat{e}(v12^{y}, g_2) = h_t^{y\alpha}$$

$$g_t^{b\alpha} h_t^{y\alpha}/h_t^{y\alpha} = g_t^{b\alpha}$$

By checking $g_t^{a\alpha} \overset{?}{=} g_t^{b\alpha}$, Bob will know whether the hidden value of $[b]$ passed the verification test or not. Since α is unknown to Bob, so bob will not learn anything about a.

Up to this point the protocol will only work properly if Alice is honest when generating the transcripts. To make the protocol secure against malicious provers, we have to do a few more things.

Active Security：

To guard against dishonest prover, the protocol would require Alice to provide three additional transcripts: $p_\alpha = {g_2}^{\alpha^{-1}}, Sig_{h_t}, Sig_{g_t}$

First, verify that $[a'], [b']$ are created from $[a], [b]$ and the same secret α.

$$\hat{e}([a'], p_\alpha) = g_t^a h_t^x \stackrel{?}{=} e([a], g_2)$$

$$\hat{e}([b'], p_\alpha) = g_t^b h_t^y \stackrel{?}{=} e([b], g_2)$$

Second, verify that v12 is created from the same secret α.

$$\hat{e}(v12, p_\alpha) = e\left(h^\alpha, {g_2}^{\alpha^{-1}}\right) \stackrel{?}{=} h_t$$

Third, confirm v11 is created from base point h_t exponent $a\alpha$.

$$PK_{g_t} = \frac{e([a'], g_2)}{v11} = \frac{g_t^{a\alpha} h_t^{x\alpha}}{h_t^{x\alpha}} = g_t^{a\alpha}$$

after retrieve PK_{g_t}, verifier can use Sig_{g_t} to prove the knowledge of $a\alpha$ on g_t.

After plugging the fact-hiding sub-protocol to the ZK protocol for arithmetic equation, Bob no longer needs to send its data facts to be verified in clear text to Alice, and Alice the prover no longer need return anything other than a Boolean answer back to bob.

Fact-hiding protocol is not limited to the above algorithm which only compares whether the hidden values of two commitments equal. More advanced algorithms that support comparison operators (e.g. $[a] > [b]$) is also available. We will not cover them in this report because they are a bit too complicated to be included in this report.

3.4 3D Zero Knowledge Protocol Review

3D Zero Knowledge Protocol was born in 2017. At the time SNARKs algorithms were largely non-universal (require expensive preprocessing for every application) and lack-transparency (require trusted setup). Although latest SNARKs have made much progress on these issues, they are still very expensive to use and some even rely on dangerously forward-thinking security assumptions (e.g. Group of Unknown Order). On the other hand, the trivial approach has largely been ignored and therefore hasn't made much progress at all. Nevertheless, 3D ZK Protocol designed three years ago is not perfect either, it still has a lot of short comings:

1. To achieve transparency, the protocol must apply Type 1 or Type 2 pairing, not the most efficient and arguably most popular Type 3 pairing.
2. Although prover cost is just the creation of a digital signature and the creation of Pedersen Commitments (one group exponential operation each since the hidden value is cached), Verification cost is linear and on each multiplication operation is approximately one pairing, which is on the expensive side.
3. When Type 3 pairing is used in a blockchain consortium, the consortium would need to use a MPC protocol to set up the base point h_1 and h_2. MPC approach usually just require one participant to be honest for the whole protocol to be safe. However, MPC based initiation approach will make cross blockchain data connection almost impossible because it is almost guaranteed that different consortium will generate different initiation parameters.

The performance of this version of 3D ZKP largely dependents on the implementation the pairing function. In practice we use the pairing parameters defined in China's SM9 standard that runs on R-ate pairing. The table below is the average performance tested on a single core Intel i5 @2.5Ghz CPU.

Unit：ms	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_t$
Point Add （One）	0.006ms	0.008ms	0.003ms
Point Add （Batch）	0.0007ms	0.001ms	
Point Mult.（Base）	0.018ms	0.054ms	0.131ms
Point Mult.（Random）	0.068ms	0.155ms	0.473ms
Pairing	0.508ms		

The average prover cost of a multiplicative operation per thread is approximately 0.02ms, and the average verifier cost is approximately 0.6ms. Note that multiplicative operation defined here is not the same as the "multiplication gate" defined in SNARKs protocols, whereas a constraint system (e.g. R1CS) is usually used to represent arithmetic operations. Constraint system not only double the number of multiplication gates required to capture an arithmetic operation, it often require expensive application level pre-processing (translate an arithmetic circuit to a SNARKs protocol understandable format), something almost never captured in their performance comparison table.

Section Four: FXN, The Next Generation 3D ZK Protocol.

FXN (stands for Fact-hiding Cross-validation Network) is the replacement protocol for the algorithm explained in section three. Although it has no relationship with the original protocol, it is compatible with all existing blockchain networks using Pedersen Commitment to encrypt data.

FXN is magnitudes more efficient than the original protocol. It not only retains the lowest prover cost known (1n EXP), but also offers one of the cheapest verifier cost known when operating in fast-verifier mode.

FXN runs in two operation modes. The standard mode offers the lowest prover cost known of just 1n EXP. The verification cost is also just around 1n EXP but still grows linearly. In the fast verifier mode, the prover cost is slightly increased to approximately 3n EXP but the verifier cost is further dropped to just 9 EXP. FXN paper will be released in 2021.

The performance comparison chart of FXN and other popular SNARKs and ZK protocols are listed below. To make comparison easier, we set 30EXP = 1Pairing. Furthermore, we lower Bulletproof's performance by three times since it cannot use base point to speed up elliptic curve point multiplication due to its use of Pedersen Vector commitment (require *n* unique base point g).

Name	Transparency	SRS Universal	Circuit Universal	Prover work cost	Verifier work cost
Groth16	○	○	○	≈5n EXP (4n + m – 1 EXP)	≈122 EXP (3P, l EXP)
Bulletproof	●	●	○	≈15n EXP (5n EXP)	≈21n EXP (7n EXP)
Sonic	○	●	○	273n EXP	≈390 EXP (13P)
PLONK	○	●	○	≈22n EXP (11n+11a EXP)	≈78 EXP (2P, 18 EXP)
FXN (Standard)	●	●	●	**1*n* EXP**	**≈1n EXP**
FXN (Fast Verifier)	●	●	●	**≈3*n* EXP**	**9 EXP**

FXN no longer needs pairing and only depends on discrete logarithmic assumption. We realized some latest SNARKs rely on new and/or questionable security assumptions such as group of unknown orders to achieve transparency. Such radical approach may sound plausible in academia research, but it could be dangerous for industry adaption.

Other than being universal and transparent, the practical performance of FXN is even better than the chart suggests. This is because FXN process proof generation and verification from the arithmetic equations directly, unlike SNARKs where an arithmetic equation first needs to be interpreted to and captured by a constraint system and then converted to a format processable by the underlying algorithm (a process sometimes denoted as “circuit preprocessing”). The only arguable down side of FXN is that the space consumption is not technically succinct. However, real life experience tells us that succinctness here is not needed because the proof transcripts should only be known by the validation requester (verifier) and the data owner (prover). Such proof transcripts shall never be stored on a public storage (e.g. blockchain) nor be validated by anyone other than the validation requester.

Section Five: Secure Multi-Party Computation

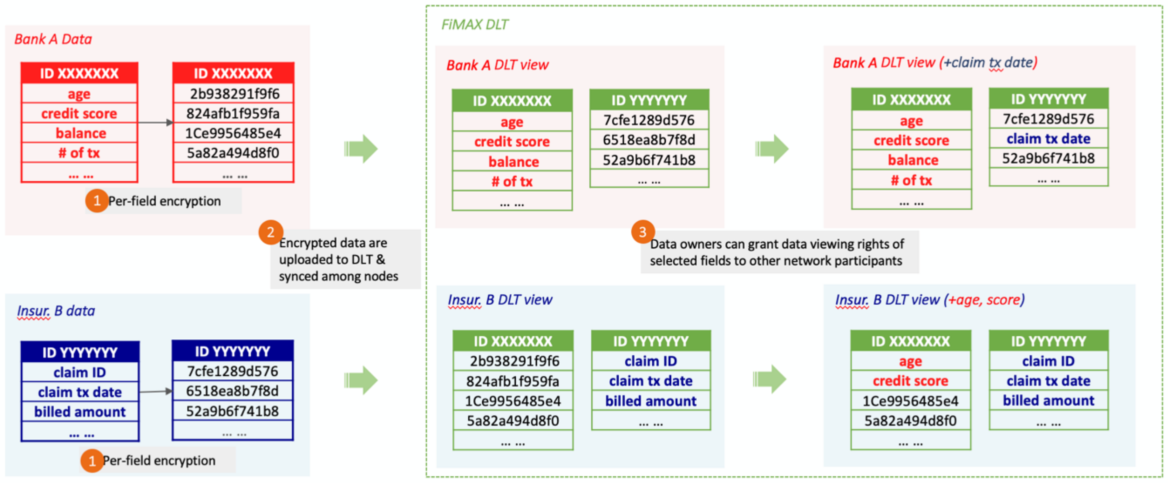
We believe secure computation is the future for enterprise blockchain because data is what make businesses thrive in the information age.

Since Blockchain itself is a decentralized storage system, it can be easily converted to a system that enables advanced federated learning. In this setting, each participant will encrypt their data and store them on a blockchain, and then leverage different secure computation technologies to securely process its data with other’s data.

In our example we present a case where an insurance company and a bank agree to collaborate and use each other’s data to build a data model while still keeping their own data private.

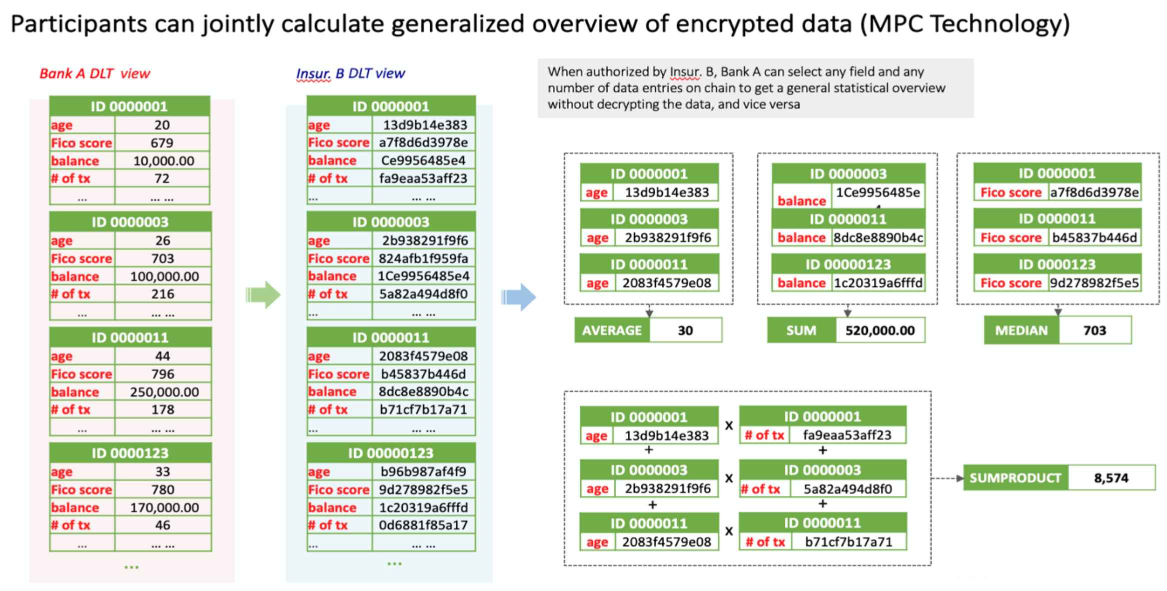
The first step is to have both participants use key derivation trick to encrypt every data field with a unique secret key (per-field encryption) and put data on a blockchain. This step alone will give participants granular access control of data at field level, allow participants to grant others access to its data at field level.

Data owner can grant data access rights of selected fields to designated parties



To enable secure computation, we require all numeric data to be encrypted using Pedersen Commitment. Note that this is also a pre-requisite for the 3D ZKP/FXN protocol explained above. Commitment schemes are building blocks for many MPC protocols. Once we have a standardized the encryption protocol, it is almost straight forward to plug in known MPC algorithms serve your need.

In this example, Bank A and Insurance Company B joined their data and use MPC protocols to calculate average age, total sum of account balances, median fico score, and sum product of age-transaction count of a group of customers while keeping everyone’s data secure, allowing more data feed into their learning model.


Participants can jointly calculate generalized overview of encrypted data (MPC Technology)

When authorized by Insur. B, Bank A can select any field and any number of data entries on chain to get a general statistical overview without decrypting the data, and vice versa

Bank A DLT view

ID 0000001	
age	20
Fico score	679
balance	10,000.00
# of tx	72
…	… …

ID 0000003	
age	26
Fico score	703
balance	100,000.00
# of tx	216
…	… …

ID 0000011	
age	44
Fico score	796
balance	250,000.00
# of tx	178
…	… …

ID 0000123	
age	33
Fico score	780
balance	170,000.00
# of tx	46
…	… …

…

Insur. B DLT view

ID 0000001	
age	13d9b14e383
Fico score	a7f8d6d3978e
balance	Ce9956485e4
# of tx	fa9eaa53aff23
…	… …

ID 0000003	
age	2b938291f9f6
Fico score	824afb1f959fa
balance	1Ce9956485e4
# of tx	5a82a494d8f0
…	… …

ID 0000011	
age	2083f4579e08
Fico score	b45837b446d
balance	8dc8e8890b4c
# of tx	b71cf7b17a71
…	… …

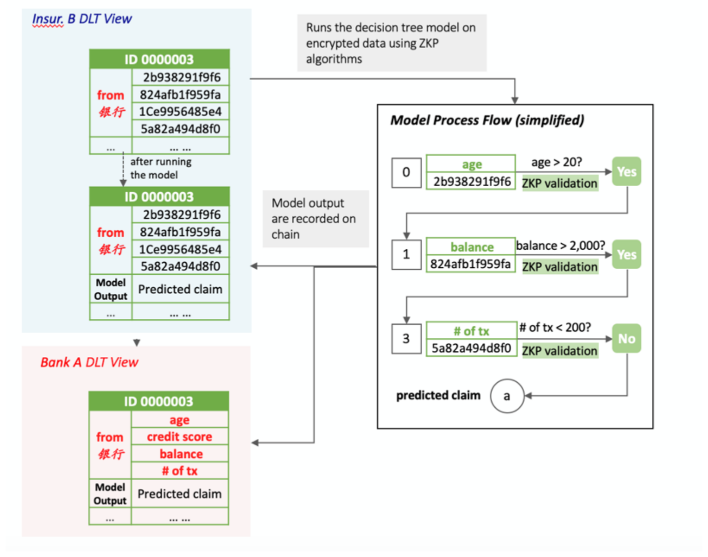
ID 00000123	
age	b96b987af4f9
Fico score	9d278982f5e5
balance	1c20319a6fffd
# of tx	0d6881f85a17
…	… …

…

ID 0000001 age 13d9b14e383
ID 0000003 age 2b938291f9f6
ID 0000011 age 2083f4579e08
AVERAGE 30
ID 0000003 balance 1Ce9956485e
ID 0000011 balance 8dc8e8890b4c
ID 00000123 balance 1c20319a6fffd
SUM 520,000.00
ID 0000001 Fico score a7f8d6d3978e
ID 0000011 Fico score b45837b446d
ID 0000123 Fico score 9d278982f5e5
MEDIAN 703
ID 0000001 age 13d9b14e383 x ID 0000001 # of tx fa9eaa53aff23
+ +
ID 0000003 age 2b938291f9f6 x ID 0000003 # of tx 5a82a494d8f0
+ +
ID 0000011 age 2083f4579e08 x ID 0000011 # of tx b71cf7b17a71
SUMPRODUCT 8,574

after a learning model on customer risk analysis is built, Insurance Company B can run its data model on data of both companies. For example, Insurance Company B model's decision tree may require data from Bank A to complete risk analysis of a customer. Instead of giving Insurance Company B access to the actual data, Bank A use ZKP protocols to return Boolean answers to Insurance Company B's data model to complete the analysis.

Running model on encrypted data using efficient ZKP algorithms



Section Six: Key Patents

201910382454.2 Zero Knowledge Technology for Trade Finance Transaction Management; Frank Lu; Menghan Wang; Dayue Zhao; Bao Zhang

202010068608.3 Zero-knowledge proof method, device and storage medium；Frank Lu; Xuejia Lai; Mu Jia; Danli Xie; Pengcheng Zhang

201910390795.4 Blockchain based transaction processing method and device; Pengcheng Zhang; Mu Jia; Frank Lu

201811529043.3 A root key protection and access system for computer readable

storage medium and system; Chengyong Feng; Frank Lu; Songsong Zhang

202010326385.6 A method to compare encrypted values for device and storage medium；Frank Lu; Xuejia Lai; Mu Jia; Pengcheng Zhang; Danli Xie

201910542027.6 A proof system for size comparison of encrypted data；Danli Xie; Wenming Zhang; Mu Jia; Frank Lu

201810874378.2 Cross-ledger transaction method and device; Frank Lu; Xuejia Lai; Mu Jia; Danli Xie

201810436870.1 User communication method, device, and storage medium on blockchain；Mu Jia; Danli Xie; Frank Lu

201910885396.5 Data calling method, device and computer readable storage medium； Pengcheng Zhang; Mu Jia; Frank Lu; Danli Xie; Fuqiang Jiang; Xiaoli Zhang

201811250449.8 Device, method and storage medium for blockchain transaction processing; Frank Lu; Zhenfei Chu; Shiwei Feng

201910371485.8 A blockchain consensus method and storage medium; Zhenfei Chu; Peipei Zhang; Shiwei Feng; Wenqiang Li

202010354819.3 Blockchain based data processing method, device, and storage medium; Bao Zhang; Danli Xie; Menghan Wang; Bin Zhu; Enke Liu

201910185053.8 Device, method and storage medium for endorsement in blockchain; Zhenfei Chu; Wei Zhang; Peipei Zhang; Wenqiang Li; Yujian Zhang

202010046528.8 Transaction endorsement processing method, server and computer readable storage medium; Frank Lu; Muhao Chen; Shiwei Feng; Zhenfei Chu

201910435849.4 Data encryption method, device, computer equipment and storage medium; Menghan Wang; Dayue Zhao; Luyan Zha; Zhuoxin Yi

201910671356.0 Updating method, device, medium for data fields stored in blockchain; Pengfei Huan; Zhuoxin Yi; Danli Xie; Fei Chen

202010363964.8 Node synchronization method and storage medium of blockchain; Frank Lu; Jie Yao

201910435849.4 Data encryption method, device, and storage medium; Menghan Wang; Dayue Zhao; Luyan Zha; Zhuoxin Yi

201910667396.8 Blockchain construction method, device, medium on cloud service; Pengfei Huan; Wei Zhang; Qingliang Zhang; Songsong Zhang

201811381039.7 Method and related equipment for recovering lost secret key based on symmetric encryption; Wenming Zhang; Ruixue Wang; Danli Xie; Pengfei Huan

201710060336.0 Safe transaction method and system based on blockchain; Frank Lu; Pengfei Huan; Yu Zhang

201911150248.5 Blockchain based file access method, device, computer equipment and storage medium; Wantao He; Pengfei Huan; Yu Lu; Yang Yang

201910440458.1 Blockchain based project data verification method and device; Jianxin Gao; Jun Lai; Menghan Wang; Dayue Zhao; Enke Liu; Bao Zhang; Luyan Zha

Special Thanks to :

Zhenfei Chu, Yu Zhang, Dayue Zhao, Jianxin Gao, Wenming Zhang, Ruixue Wang, Yujian Zhang, Songsong Zhang, Cheng Feng, Bao Zhang, Wei Zhang, Wanjing Chen, Zhengxiang Fang, Huadu Li, Weilin He, Yun Huo, Guojun Luo, Yan Li, Cheng Lin, Meng Wang, Li Zhu, Yangyue Feng, Mingce Xue