The secure transmision and analysis of real-time data streams from remote distributed data sources at the edge is becoming increasingly essential to many business models and scientific workloads. Past analytical results inform future data collection and analysis in a fast iterative feedback loop. This rapidly evolving analytic pipeline, with its explicit objective of controlling real-world operations in real-time, raises a critical need for securing the entire chain from edge to data center and back. Herein we propose a *Zero-Trust*, AES, FullyHomomorphic Encryption (FHE), and TLS-based approach as a solution. In this model, data from edge operations is owned by a single organization, often originating from geographically diverse locations, is highly confidential, and must remain secure from unauthorized access and/or breach. Data in plaintext form as well as secret decryption keys can never be exposed. High volumes of data must be securely communicated over untrusted network links and securely combined and analyzed on a publicly or commercially shared untrusted aggregation server. The analytical result must then be securely communicated over an untrusted network link back to a secure, trusted location for final decryption and evaluation. The secured results of the evaluation then inform changes in the data collection parameters at the edge sensors, d so entering another iteration cycle.

**Example Use Cases:**

|  |  |
| --- | --- |
| A large round machine in a factory  Description automatically generated | **Securing large scale physics experimentation using geographically distributed sites**   * Large physics experimental installations are geographically distributed. * Securely, efficiently, and cost effectively coordinating experiments involving sensor data collection and data analysis over multiple geographically disperse data collection sites is extremely challenging using currently available approaches. |
| A fighter jet flying in the sky  Description automatically generated | **Securing the optimization of military aircraft maintenance and functionality**   * Military Aircraft Manufacturers (MAMs) require data to be collected on aircraft operational use and analyzed for both maintenance as well as functional enhancements. * National security requires that all military aircraft operational use data be secure from unauthorized access and/or breach. |
| Long shot of a truck  Description automatically generated | **Securing oil and gas exploration data and analysis**   * Local contractors with low expectation of trust are often used to collect data from remote sites. * Oil and gas exploration data and analytical results are extremely valuable and must remain confidential and secure from unauthorized access and/or breach. |

In all these cases, the data, its analytical results, and the data collection parameters must remain provably secure from unauthorized access and/or breach even when using untrusted or e4ven publicly shared network links d data center servers. The analytical results must be securely transmitted to a secure and trusted administrator who can evaluate the results d adjust the data collection parameters and remote system controls, iterating the process as necessary.

These use cases differ significantly from areas such as AdTech d Health Care maintaining secrecy and/or anonymity of Personally Identifiable Information (PII) is of critical importance. In those cases, PII data can never be exposed. Specifically, in this distributed sensor example, the data is wholly owned by a single entity, has full access authority but must leverage shared untrusted infrastructure.

**The Problem with Current Security Models**

|  |  |
| --- | --- |
| For the entire history of the computer industry, data must be available in plaintext to be processed. This requirement severely impacts data security since a perimeter defense must be constructed to prevent unauthorized access and/or breach of the underlying plaintext data. The problem with this approach is that it requires the user to trust that the computational infrastructure is not poorly designed or compromised. Perimeter defense designs are extremely difficult to prove secure as they require the infrastructure partner(s) along with the data owner to be included as part of the trust environment. They do not implement a provable Z*ero-Trust* security model. | A computer on a desk with papers and a lamp  Description automatically generated |

**Security Examples Requiring Trust Partner**

|  |  |
| --- | --- |
| To protect plaintext data being processed at a data center level, a perimeter is constructed using thousands of hardware and software components that are constantly changing through bug and security fixes, feature upgrades, and product replacements. Given the ever-changing nature of the wide variety of components supplied by a multitude of different vendors, it is a statistical certainty that breach points will always exist. The only question is whether they have been identified by adversaries. This is the very definition of a provably insecure model. Standard perimeter defense data security architectures cannot provide provable *Zero-Trust* security. | A wall of multicolored bricks  Description automatically generated |
| Current Trusted Execution Environments (TEEs) have similar issues and cannot provide a provable Z*ero-Trust* security model. When viewed by a programmer, a typical ARM/x86/AMD CPU looks like a deterministic execution engine of program instructions. Looking below that interface shows a different picture. To efficiently implement instruction execution, many supporting components must operate in asynchronous and non-deterministic fashion enabling a vast myriad of attacks that can be launched against a TEE. |  |

A recent survey of TEE security[[1]](#footnote-2) has stated that current attacks include, “*software-based attacks, side-channel attacks and (micro-) architectural attacks. Although some of these attacks are theoretical, many of them can be realized and have been exploited in practice. What is worse, countermeasures have only been developed for some of them.*”

The answer to the above vulnerabilities is to assume that infrastructure cannot be trusted and is already compromised. Provable data security can only be achieved via a *Zero-Trust* model that omits infrastructure trust. A *Zero-Trust* model is where the data is secured by encryption techniques based on currently understood to be unsolvable mathematics with no trust required from the surrounding infrastructure environment. The unsolvable mathematical challenge provides an air tight barrier to intruders rather than a statistical barrier.

**Proposed Solution**

The proposed solution requires addressing three distinct issues:

1. Perform provably secure remote sensor data transmission,
2. Perform provably secure remote sensor data aggregation and analysis, and,
3. Securely transmit system control signals based on previous analytical results.

**Secure Data Transmission**

Once data has been collected from remote sensors, it must be securely transmitted over an untrusted network link to an untrusted aggregation site for secure analysis. That means the data must be encrypted. The encryption mechanism must have a low plaintext-to-ciphertext expansion ratio and is either provably secure or have its parameters configured to be provably resistant from conventional and quantum attacks as is the case for AES-256 symmetric encryption[[2]](#footnote-3).

**Secure Data Analysis**

Once the data has been securely communicated, it must be analyzed in a provably secure manner, e.g. considered secure from conventional and quantum computer attacks. FHE is a Public Key Enabled (PKE) scheme based on currently understood to be unsolvable challenges in lattice mathematics. The US National Institute of Standards and Technology (NIST) considers methods based on these challenges to be Post Quantum Cryptography (PQC)[[3]](#footnote-4). FHE supports unrestricted computation on encrypted data producing results encrypted in the same way. Encrypted data is equivalent to noise. Plaintext data and its derivative computational results are never exposed during the computational lifecycle, even on compromised infrastructure. FHE implements a *Zero-Trust* model of data security, requires no additional trusted parties, and can only be viewed by using its protected secret decryption key.

Using FHE in this model has two challenges:

* Processing encrypted data using FHE requires significant and scalable hardware acceleration.
* FHE’s expansion ratio of two to three orders of magnitude over plaintext makes it unsuitable for high volume data transmission over untrusted Wide Area Network (WAN) links.

**Scalable Hardware Acceleration**

To make FHE a usable computer security technique, significant amount of computation must be performed in a relatively short timeframe. Leveraging Computer Science research in both Data Flow[[4]](#footnote-5) and Reconfigurable Computing[[5]](#footnote-6), Cornami has developed a hardware architecture that can scale the amount of processing and I/O to meet FHE application requirements. Existing processor architectures like CPUs and GPUs do not easily scale in processing or I/O. These architectures significantly impede a programmer’s ability to express the full parallelism and pipelining available within the mathematics of the FHE domain. Using Data Flow and Reconfigurable Computing to define a scalable fabric of processing cores allows the Cornami architecture to dial up or dial down the amount of computational parallelism and pipelining being applied to an FHE application. This allows customer to pick the performance/hardware footprint level for the specific application response time requirement up to and exceeding real-time.

**Secure AES-256 to FHE Transcipher**

The plaintext to FHE ciphertext expansion ratio of two to three orders of magnitude presents a significant challenge to transmitting encrypted data over WAN links. It is for this reason that an end-to-end FHE solution is not practical and a PQC-resistant symmetric cipher, like AES-256, is chosen for secure data transmission. However, unrestricted computation on AES-256 encrypted data is not possible and encrypted AES-256 data must be securely translated to FHE encryption before secure analysis can occur. Secure AES-256 to FHE translation can occur through a transcipher algorithm where neither plaintext data nor secret decryption keys are exposed.

**AES-256 to FHE Transcipher Algorithm**[[6]](#footnote-7)

An AES-256 to FHE transcipher is an FHE computation that takes as input two arguments:

1. An AES-256 encrypted data block, and,
2. An FHE encrypted, AES-256 secret symmetric key and transforms the AES-256 encrypted data block to an FHE encrypted data block.

It does this without ever exposing data in plaintext or secret decryption keys during its execution and can execute on an untrusted platform. The AES-256 secret symmetric key is encrypted using a previously generated FHE public PKE key.

FHE functions, by definition, can run on an untrusted platform and their input arguments require at least one FHE encrypted value. If there are multiple FHE encrypted arguments, they must all be encrypted with the same key. In addition to at least one FHE encrypted argument, non-FHE encrypted values can also be arguments. In the AES-256 to FHE transcipher case, the program performs an AES-256 decryption as an FHE function of two arguments: the AES-256 encrypted data block (being the non-FHE encrypted argument) and the FHE encrypted AES-256 secret symmetric key. This FHE computation produces the AES-256 decryption as an encrypted FHE result, encrypted in the same way as the FHE encrypted AES-256 secret symmetric key argument. As stated earlier, data in plaintext data or secret decryption keys are never exposed during processing and can safely run on an untrusted platform.

A screenshot of a computer

Description automatically generated

**Securely Updating Remote Sensor Data Collection Parameters**

A screenshot of a computer screen

Description automatically generatedTo securely update remote sensor data collection parameters, the administrator uses a PQC Transport Layer Security (TLS) protocol[[7]](#footnote-8) [[8]](#footnote-9) (done via a Secure Shell (SSH) session) to sign onto Remote Site Control Computers (RSCCs) to configure sensor data collection settings. The PQC TLS process works as follows:

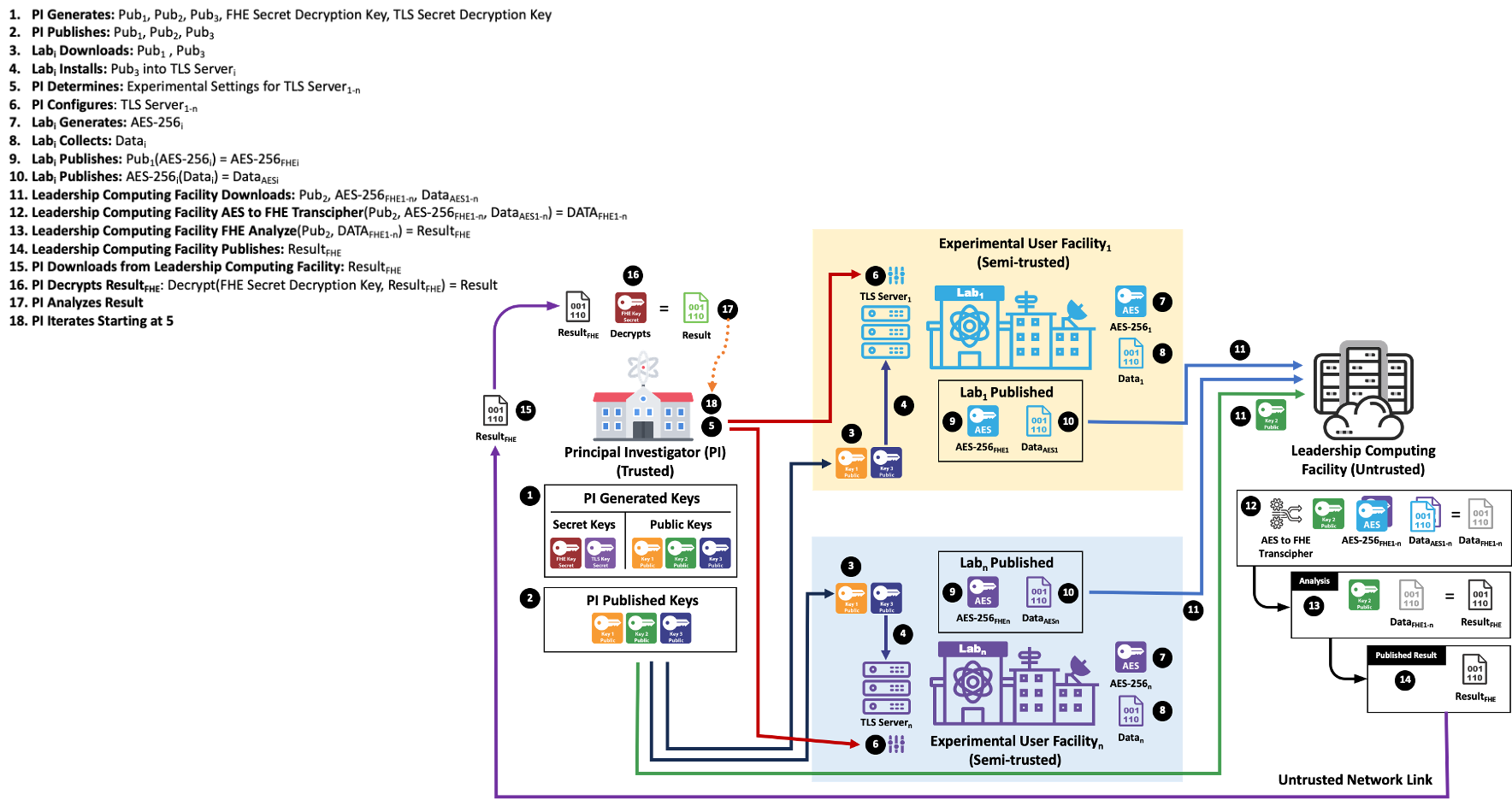
**Securely Transmitting and Analyzing Data from Remote Edge Sites**

This solution to the problem of securely transmitting and analyzing data from remote edge data collection is based on the administrator being the sole owner of the data and its analytic results; and based on securely transmitting conventional secure and quantum secure/resistant cryptography. The trusted Principal Investigator (PI) constructs the analytical model, securely defines the data to be collected, securely analyzes the data, and securely evaluates its results. The PI then determines if adjustments to data collection is required to iterate the process and, if so, securely redefines those new parameters. This approach has three functional areas:

* Trusted PI site
* Remote, untrusted data collection site(s)
* Remote, untrusted data aggregation and analysis site

Below is the functional process. Each remote data collection site is defined as Labi, each remote site control computer is defined as RSCCi, and each remote site AES-256 symmetric key is defined as AES-256i:

1. The PI generates five keys:
   1. FHE Secret Decryption Key
   2. FHE Public Encryption Key (Pub1)
   3. FHE Public Evaluation Key (Pub2)
   4. TLS Secret Decryption Key
   5. TLS Public Encryption Key (Pub3)
2. The PI publishes the three public keys:
   1. FHE Public Encryption Key (Pub1)
   2. FHE Public Evaluation Key (Pub2)
   3. TLS Public Encryption Key (Pub3)
3. Each remote data collection site (Labi) downloads Pub1 and Pub3.
4. Each Labi:
   1. Generates its own AES-256 secret symmetric key (AES-256i)
   2. Using Pub1, each Labi FHE encrypts its AES-256i: Pub1 (AES-256i)
5. Each Labi installs Pub3 into the TLS server on its RSCCi.
6. The PI initiates a PQC TLS SSH session at each RSCCi at each Labi
   1. PI securely configures data collection parameters at the RSCCi at Labi
7. Each Labi then initiates data collection.
   1. Once data collection is complete, each Labi encrypts the plaintext data with its AES-256i symmetric key.
   2. Each Labi then transmits the AES-256i encrypted data and the FHE encrypted AES-256i key to the remote, untrusted data aggregation and analysis site.
8. The remote, untrusted data aggregation and analysis site receives the AES-256i encrypted data and FHE encrypted AES-256i key from each Labi. It downloads the Pub2 from the PI.
   1. For each Labi submission, an AES-256i to FHE secure transcipher is executed using the Pub2 key resulting in an FHE encrypted data set.
   2. All Labi data sets are aggregated together as one FHE encrypted data set that is encrypted with the same FHE Pub1 key.
   3. The FHE encrypted data set is analyzed homomorphically, using the Pub2 key, and an FHE encrypted result is produced.
   4. The FHE result is sent over untrusted WAN link to the PI for secure decryption in a trusted environment, evaluation, and potential readjustment of the Labi data collection parameters.



**Demonstration Use Case**

This example use case demonstration allows a PI to iteratively perform provably secure physics experiments using one, geographically remote, large experimental physics machine (SLAC National Accelerator Lab – SLAC) with computer controlled (Stepper Motor Control Computer – SMCC), stepper motor configured, sensors. Trivium symmetric key encryption[[9]](#footnote-10) is used as a place holder instead of AES-256 as a Trivium to FHE transcipher implementation was readily available to the developers. Production systems will use AES-256 symmetric key encryption.

1. The SLAC S2AI SANDBOX FOR STREAMING AI, Principal Investigator (PI) generates five keys:
   1. FHE Secret Decryption Key
   2. FHE Public Encryption Key (Pub1)
   3. FHE Public Evaluation Key (Pub2)
   4. TLS Secret Decryption Key
   5. TLS Public Encryption Key (Pub3)
2. The PI publishes the three public keys:
   1. FHE Public Encryption Key (Pub1)
   2. FHE Public Evaluation Key (Pub2)
   3. TLS Public Encryption Key (Pub3)
3. SLAC S3DF SLAC SHARED DATA SCIENCE DATA FACILITY (S3DF) downloads Pub1 and Pub3 over untrusted network link
4. S3DF:
   1. Generates its own Trivium symmetric key (Trivium). In production AES-256 will be used being used. Trivium is a place holder for AES-256 in this example.
   2. Using Pub1, FHE encrypts its Trivium symmetric key: Pub1 (Trivium)
5. S3DF installs Pub3 into the TLS server on its SMCC.
6. The PI initiates a PQC TLS SSH session on the SMCC
   1. The PI securely configures data collection parameters at S3DF
7. S3DF initiates data collection.
   1. S3DF data collection consists of a plaintext matrix.
   2. Once data collection is complete, S3DF encrypts the plaintext matrix data with its Trivium symmetric key.
   3. S3DF then transmits the Trivium encrypted data and the FHE encrypted Trivium symmetric key over untrusted WAN link to the Oak Ridge National Laboratory Data Center (Oak Ridge) for untrusted data aggregation and analysis.
8. The Oak Ridge downloads Pub2 from the PI and receives the Trivium encrypted data and FHE encrypted Trivium key from S3DF.
   1. A Trivium to FHE secure transcipher[[10]](#footnote-11) is executed on the Trivium encrypted S3DF data set resulting in an FHE encrypted data set.
   2. An addition is performed homomorphically, using the Pub2 key, to the FHE encrypted matrix data producing an FHE encrypted result. In the production form this will be replaced by an arbitrary computation.
   3. The FHE result is sent over untrusted WAN link to the PI for secure decryption in a trusted environment, evaluation, and potential readjustment of the S3DF SMCC data collection parameters.

A diagram of a network link

Description automatically generated

**Summary**

The secure collection, transmission, evaluation, and adjustment of data collection parameters is a general use case of value to government, academia, and business. Current security models are not provably secure because they all require trust in the data processing infrastructure. A *Zero-Trust* approach, based solely on ciphers based on mathematics challenges currently understood to be unsolvable or resistant to quantum attack is the only way to achieve a provable security model for distributed data collection, analysis, and data collection parameter adjustment. A solution template has been provided based on quantum resistant AES-256 and PQC FHE and TLS cryptographies. Along with this template, a simple Proof of Concept demonstration using Trivium as a place holder for AES-256 has also been presented. Separating data access from data processing is the key concept in achieving the benefits of this model. By accelerating the analytical computation of the FHE encrypted data, reducing data transmission overhead by first collecting data in AES-256 and securely transciphering to FHE, and allowing for a secure TLS session to adjust data collection parameters, this model enables a provably secure PQC mechanism for iterative distributed data collection and analysis.

1. Antonio Muñoz, Ruben Ríos, Rodrigo Román, Javier López, A survey on the (in)security of trusted execution environments, Computers & Security, March 14, 2023, Elsevier – https://www.sciencedirect.com/science/article/pii/S0167404823000901?via%3Dihub [↑](#footnote-ref-2)
2. Quantum Security Analysis of AES, Xavier Bonnetain, María Naya-Plasencia, and André Schrottenloher – https://eprint.iacr.org/2019/272.pdf [↑](#footnote-ref-3)
3. FHE-Related Comments on NIST First Call for MultiParty Threshold Schemes, Yuriy Polyakov, MPTS 2023: NIST Workshop on Multi-Party Threshold Schemes 2023, September 2 7, 2023 – https://csrc.nist.gov/csrc/media/presentations/2023/mpts2023-day2-talk-fhe-comments/images-media/mpts2023-2a1-slides--yuriy--FHE-comments.pdf [↑](#footnote-ref-4)
4. Data Flow Wikipedia – https://en.wikipedia.org/wiki/Dataflow [↑](#footnote-ref-5)
5. Reconfigurable Computing Wikipedia – https://en.wikipedia.org/wiki/Reconfigurable\_computing [↑](#footnote-ref-6)
6. Nir Drucker, A note about Authenticated Transciphering: decrypting AES under Homomorphic Encryption using CKKS, NIST Information Technology Laboratory, COMPUTER SECURITY RESOURCE CENTER, MPTS 2023: NIST Workshop on Multi-party Threshold Schemes 2023, September 27, 2023 – https://csrc.nist.gov/Presentations/2023/mpts2023-day2-talk-fhe-aes-transcipher [↑](#footnote-ref-7)
7. NIST Announces First Four Quantum-Resistant Cryptographic Algorithms, July 05, 2022 – https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms [↑](#footnote-ref-8)
8. https://github.com/pq-crystals/kyber [↑](#footnote-ref-9)
9. Trivium Specifications, Christophe De Cannière and Bart Preneel, October 20, 2016 - https://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium\_p3.pdf [↑](#footnote-ref-10)
10. Trivial Transciphering With Trivium and TFHE, Thibault Balenbois, Jean-Baptiste Orfila, and Nigel P. Smart, September 21, 2023 - https://eprint.iacr.org/2023/980.pdf [↑](#footnote-ref-11)