THE RISE OF CONFIDENTIAL COMPUTING

Mark Russinovich
Azure CTO
Microsoft
Cloud Data Threats

Customer cloud data concerns:

- Malicious privileged admins or insiders
- Hackers exploiting bugs in the Hypervisor/OS of cloud fabric
- Third parties accessing it without customer consent

CSA’s Cloud Computing Top Threats
February 2016

Top Threat: Data Breaches
Data Protection

At rest
Encrypt inactive data when stored in blob storage, database, etc.

Examples:
- Azure Storage Service Encryption for Data at Rest
- SQL Server Transparent Database Encryption (TDE)

In transit
Encrypt data that is flowing between untrusted public or private networks

Examples:
- HTTPS
- TLS

In use
Protect/Encrypt data that is in use during computation

Examples include:
- Trusted Execution Environments
- Homomorphic encryption
Trusted Execution Environments (TEEs)

**Protected container:**
- Isolated portion of processor & memory
- Code & data cannot be viewed or modified from outside

Supports attestation: proving of identity both locally and remotely
Supports sealing: persisting secrets

**Examples:**
- Intel SGX
- Virtualization Based Security (VBS) aka Virtual Secure Mode
Intel SGX (Software Guard Extensions)

**SGX goal:** Minimize hardware attack surface

Instructions to set aside private regions ("protected containers") of code and data

Data is only in clear within protected memory and CPU
TEE application architecture
Hyper-V Virtualization Based Security (VBS)*

Virtual trust layers (VTLs) identify access levels

VTL 1 defines Isolated User Mode (IUM)

VBS prevents the OS, applications in VTL 0 and device drivers from accessing IUM (VTL 1)

*AKA Virtual Secure Mode (VSM)
## TEEs compared to other secure hardware

<table>
<thead>
<tr>
<th></th>
<th>TPMs</th>
<th>HSMs</th>
<th>TEE (Intel SGX)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware</strong></td>
<td>Separate physical chip; embedded into motherboard</td>
<td>Separate external device</td>
<td>Built into CPU</td>
</tr>
<tr>
<td><strong>Operations</strong></td>
<td>Secure cryptographic operations</td>
<td>Secure cryptographic operations*</td>
<td>Secure “container” in which to run arbitrary code</td>
</tr>
<tr>
<td><strong>Attestation</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Sealing</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Use case</strong></td>
<td>Local system integrity</td>
<td>Key management system</td>
<td>Generalized compute</td>
</tr>
<tr>
<td></td>
<td>• Validate properties at boot</td>
<td>• Generate &amp; manage keys</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Host measurement</td>
<td>• Key exchange</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Platform device authentication</td>
<td>• Encryption</td>
<td></td>
</tr>
<tr>
<td><strong>Example application</strong></td>
<td>BitLocker Disk Encryption</td>
<td>KeyVault</td>
<td>SQL Always Encrypted</td>
</tr>
</tbody>
</table>

*Some permit arbitrary code, but not optimized for general purpose compute*
Common TEE application patterns

- Protect data confidentiality and integrity on remote machine
- Protect sensitive algorithmic IP (e.g. financial trade algorithms)
- Code access security, including remote clients
- Create a trusted network of nodes among a set of untrusted parties
- Centrally combine different data sources for a better algorithmic outcome without loss of confidentiality
- Protect communication with other secure device endpoints (e.g. local peer to remote HSM)
- Secure licensing and DRM
Azure Confidential Computing
Confidential cloud

Data is fully in the control of the customer regardless of whether in rest, transit, or use even though the infrastructure is not.

The cloud platform provider is outside the trusted compute base.

Code running in cloud is protected and verified by the customer.
Azure and confidential computing

Working with silicon partners to enable Confidential Computing

Building software to deploy, manage, and develop secure TEE applications on Azure

Designing and developing services to support attestation in the cloud

Enabling confidential PaaS and SaaS services
The ACC development environment

Universal
Generalize enclave application
model to minimize hardware/
software specific concepts

Pluggable
Componentization to
support desired runtimes
and crypto libraries

Standardized
Remove hardware vendor
specific signing and
verification requirements

Multi-platform
Design with all software platforms,
Windows and Linux, in mind

Compatible
Easier enablement of
redistributable applications

Open
Open source and a standard for
secure enclave-based application
Universal cloud attestation

1. Quote provide proofs:
   - Code runs in genuine SGX enclave
   - Enclave version and owner is as expected
   - Arbitrary enclave supplied data is as expected
2. Attestation service attests to hardware, rooted in Intel chain of trust, and issues token
3. Cloud service (e.g. AKV) is presented with token with certs chained to CPU
4. Token is used to release secrets to the application enclave
Preventing direct information leaks

⚠️ **Problem:** code in enclaves may unintentionally write secrets out

✅ **Solution:** use a compiler that instruments memory accesses & verify that instrumented binary does not leak secrets

---

**Guarantee:** attacker can only observe encrypted communication

---

Microsoft
Preventing indirect information leaks

⚠️ **Problem:** memory/disk access patterns may leak information

✅ **Solution:** use compiler and hardened libraries that prevent leaks with data oblivious primitives

---

**Binary decision tree**

- **Female?**
  - Yes
    - **Diabetes in family**
    - No
      - Heart disease: No
    - Yes
      - **Diabetes in family**
      - No
        - > 35
          - Diabetes in family
        - Yes
          - > 35

---

**Memory**

**Accesses from 2 predictions**

- A
- B
Demo:
Oblivious computing
Example confidential computing scenarios
Always encrypted storage with SQL Server

Enabling scalable and confidential blockchain networks with Coco Framework

Financial data processing

Secure multi-party machine learning
SQL Always Encrypted

Protects sensitive data in use from high-privileged yet unauthorized SQL users both on-premises and in the cloud.

Current GA version in SQL Server 2016/17 and Azure SQL DB

- **Client side Encryption**: Client-side encryption of sensitive data using keys that are *never* given to the database system.
- **Encryption Transparency**: Client driver transparently encrypts query parameters and decrypts encrypted results.
- **Queries on Encrypted Data**: Support for equality comparison, including join, group by and distinct operators via deterministic encryption.
Confidential SQL Always Encrypted

Protects sensitive data in use while preserving rich queries and providing in-place encryption.

Secure computations inside SQL Enclave
SQL Server Engine delegates operations on encrypted to the SQL Enclave, where the data can be safely decrypted and processed.

Rich Queries
pattern matching (LIKE), range queries (<, >, etc.), sorting, type conversions, support for non-bin2 collation, and more.

In-place Encryption
SQL Enclave can perform initial data encryption and key rotation, without moving the data out of the database.
Coco Framework: Confidential Consortium Blockchain Framework

Open-source framework that enables high-throughput (~100x), fine-grained confidentiality, and consortium governance for blockchain.

Creates a trusted network of physical nodes on which to run a distributed ledger, providing secure, reliable components for the protocol to use.

Through the use of TEEs able to simplify consensus and transaction processing.
Coco Framework architecture

Validating Node (VN)

COCO Interface (Host)

Enclave in TEE

COCO State

Adapter

Blockchain Core

Replicated Persistent Store

Microsoft
Coco Framework: Confidentiality model for Ethereum

Coco FX disables access to transaction and block level information

Ethereum smart contract enforces access control rules by verifying address of caller
// Only admins can authorize other users or admins

function addAuthorizedReader(address a) onlyAdmins {
    allowedReaders[a] = true;
    userAuthorized(a);
}

function addAuthorizedWriter(address a) onlyAdmins {
    allowedWriters[a] = true;
    userAuthorized(a);
}

function addAdmin(address a) onlyAdmins {
    admins[a] = true;
    userAuthorized(a);
}

function removeAuthorizedReader(address a) onlyAdmins {
    allowedReaders[a] = false;
    userForbided(a);
}

function removeAuthorizedWriter(address a) onlyAdmins {
    allowedWriters[a] = false;
    userForbided(a);
}

function removeAdmin(address a) onlyAdmins {
    admins[a] = false;
    userForbided(a);
}
pragma solidity ^0.6.0;

/* A simple contract that restricts access to a map of values. A user is represented by
  * an Ethereum address they own.
  */

contract Vault {
    // Each access is mapped and visible only to authorized users
    event vaultReadAccess(address byWho);
    event vaultWriteAccess(address byWho);
    event userAuthorized(address who);
    event userForbidden(address who);

    // The map that is access restricted and the lists of users that are permissioned to
    // read and write. Note that these members are not public and can only be accessed
    // by code in this contract
    mapping (bytes32 => string) secretsMap;
    mapping (address => bool) allowedReaders;
    mapping (address => bool) allowedwriters;
    mapping (address => bool) allowedEventwriters;

    // Admins can authorize users and add other admins
    mapping (address => bool) admins;

    // Check
    function check(bytes32 hash, uint8 v, bytes32 r, bytes32 s, bytes32 topic) returns (bool){
        address sender = ecrecover(hash, v, r, s);

        bytes32 eventTopic = keccak256("vaultReadAccess(address)");
        bytes32 event2Topic = keccak256("vaultWriteAccess(address)");
        bytes32 event3Topic = keccak256("userAuthorized(address)"EZ);
        bytes32 event4Topic = keccak256("userForbidden(address)"EZ);

        if (topic == eventTopic)
            // allowedReaders can access event: vaultReadAccess
            return allowedReaders[sender];
        else if (topic == event2Topic)
            // allowedwriters can access event: vaultWriteAccess
            return allowedwriters[sender];
        else if (topic == event3Topic || topic == event4Topic)
            // admins can access events: userAuthorized, userForbidden
            return admins[sender];
        else {
            throw;
        }
    }

    // Contract is created with an initial set of authorized users who can read, write and view events
    function Vault(address initialAdmins) initialAdmins)
    {
        for (uint i = 0; i < initialAdmins.length;i++){
            admins[initialAdmins[i]] = true;
        }
    }

    // A read call made via eth.call is not signed so we require the caller supply an
    // ECDSA signature to authenticate themselves
    modifier onlyAllowedReaders(bytes32 hash, uint8 v, bytes32 r, bytes32 s)
    {
        address reader = ecrecover(hash, v, r, s);
        if (!(allowedReaders[reader])
            throw;
        }
    }

    modifier onlyAllowedWriters
    {
        if (!(allowedwriters[msg.sender])
            throw;
        }
    }

    modifier onlyAdmins
    {
        if (!(admins[msg.sender])
            throw;
        }
    }

    // Modifier onlyAllowedReaders ensures only users authorized to read can call this function
    function read(bytes32 hash, uint8 v, bytes32 r, bytes32 s, bytes32 topic) constant returns(bytes32)
    {
        vaultReadAccess(msg, sender); // Generate an event for notification
        return secretsMap[key];
    }

    // Modifier onlyAllowedWriters ensures only users authorized to write can call this function
    function write(bytes32 key, string value) onlyAllowedWriters
    {
        secretsMap[key] = value;
        vaultWriteAccess(msg, sender); // Generate an event for notification
    }
Demo: Coco Ethereum versus Ethereum
Azure HDInsight

Process massive amounts of data using open source frameworks such as Hadoop, Spark, Hive, Kafka, etc.

Input
- Code
- Data

Hadoop Framework
- Map()
- Reduce()
- Data

Host Operating System

Output

Azure Storage

Microsoft

RSA Conference 2018
void Mapper::map(string k, string v) {
    /* ... */
}

void Reducer::reduce(string k, vector<string> v) {
    /* ... */
}

Inside enclave/Trusted
Secret user code
Public generic code
Protocols
5,500 LLOC

Outside enclave/Untrusted
Create enclave
Talk to OS
Bind to Hadoop

Microsoft

RSA Conference 2018
Azure Machine Learning (ML)

**Machine Learning Studio**: UI web designer to create an end-to-end ML workflow

- Large library of predictive analytic algorithms from which to train models
- Modules to support data input, output, pre-processing, and visualizations

![Diagram of ML workflow]

**Apply ML algorithm**

- Anomaly detection, classification, clustering, recommendation, regression, statistical functions, etc.

**Machine Learning API service**: service to deploy prepared ML models at cloud scale and availability
Confidential multi-party machine learning

Partnered health facilities contribute private patient health data sets to train a ML model.

Each facility only sees their respective data sets (aka no one, not even cloud provider, can see all data or trained model, if necessary).

All facilities benefit from using trained model.
Demo:
Confidential multi-party ML
Confidential computing in the cloud is in its early stages.

**Microsoft** is driving the direction & adoption of newer trusted execution environments in the cloud.

**Azure** is empowering new secure business scenarios in the cloud.
References

Blockchain with Coco Fx:
http://aka.ms/cocopaper

Multi-party machine learning:

SQL Server with Haven:

Map/reduce with VC3:

Preventing enclave information leaks:
https://people.eecs.berkeley.edu/~rsinha/research/pubs/pldi2016.pdf

Using side-channel page faults to extract JPG images: