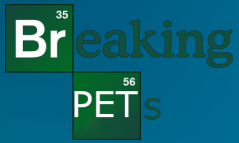
Fraunhofer AISEC



AISEC Retreat, 2026

Overview

Mode

OR

- Introduction (5min)
- PETs Overview (8min)
- 5× Attacks (5 × 20min)
  - 2min mission briefing
  - 10min 👉 YOU run the attack
  - 8min explanation
- Conclusion (7min)

= 115min

- Introduction (5min)
- PETs Overview (8min)
- 5× Attacks (5 × 20min)
  - 2min mission briefing
  - 3min 👉 YOU think about it
  - 7min 🤝 we do it together
  - 8min explanation
- Conclusion (7min)

= 115min

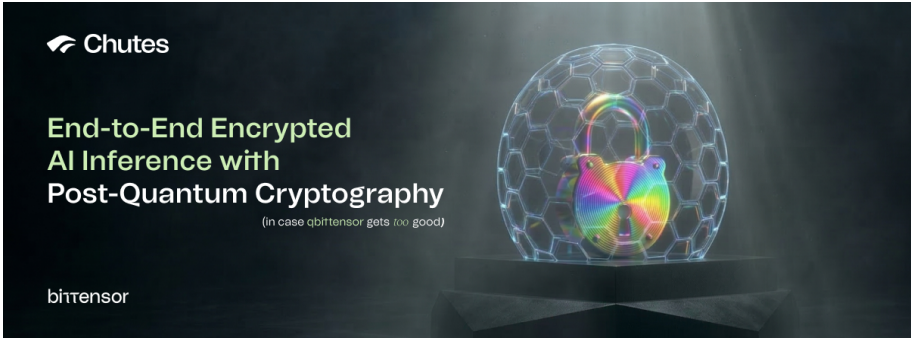
Material:

https://aisec.bramm.xyz/

Introduction

PETs everywhere.

You've seen these headlines. You've heard these pitches.



PETs everywhere.

You've seen these headlines. You've heard these pitches.

PETs everywhere.

You've seen these headlines. You've heard these pitches.

PETs everywhere.

You've seen these headlines. You've heard these pitches.

What exactly is protected here ?

What exactly is protected here ❓

(Take a moment. Think carefully.)

That's not a trick question.

It's the *only* question.

If you can't answer:

- You can't evaluate claims
- You can't find the gaps
- You can't break it

If you can answer:

- You spot the buzzword fog
- You find the attack surface
- You can **break PETs**

"Encrypted" does not mean:

Nothing leaks.

Core Thesis

PETs protect:

- computations
- plaintexts
- inputs
- statistical identities

But usually NOT:

- metadata
- timing
- communication graphs
- outputs
- access patterns
- query patterns
- volume patterns
- result patterns
- update/state patterns
- operational behavior

PETs

Differential Privacy

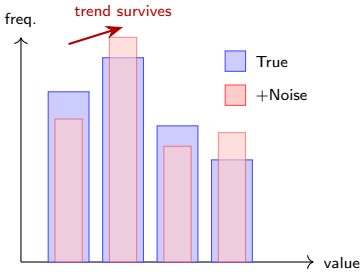
Concept [DR14]

Add calibrated **statistical noise**:

- **Enough** to hide individual records
- **Not too much** to keep aggregates useful

Leakage: Trends

Strong sub-group trends can survive the noise and still be inferred from query results



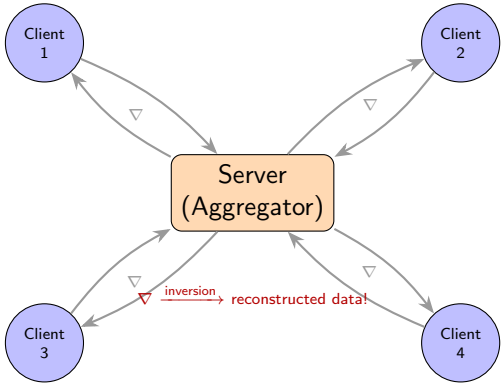
Federated Learning

Concept [MMR+17]

- Many **clients** train locally on their own data
- Only **gradients** (∇) are shared – not raw data
- A central server **aggregates** the updates

Leakage: Gradient Inversion Attack

- Shared gradients can be used to **reconstruct original training data**
- Raw data never leaves the client – yet privacy is violated



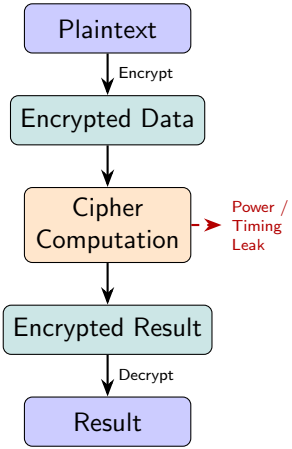
(Fully) Homomorphic Encryption

Concept [Gen09]

- Computation on **encrypted data** without decrypting it
- Result, once decrypted, equals plaintext computation

Leakage: Side-Channel Attacks

- Exploits *inadvertently leaked* information (e.g. power consumption, timing)
- **Access patterns** can be inferred



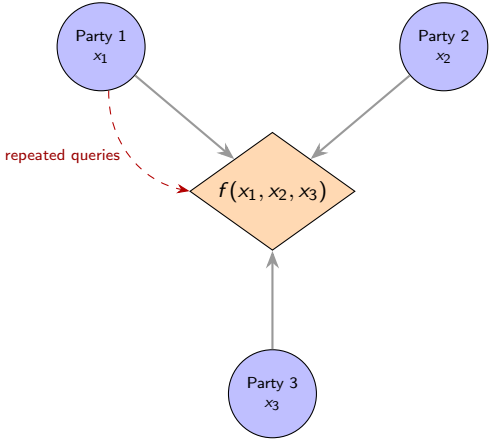
Multi-Party Computation

Concept [Yao82]

- Parties **jointly compute** $f(x_1, x_2, \ldots, x_n)$
- Each party keeps their **input private**
- Only the **final result** is revealed

Attack: Public Anchor Leakage

Repeated queries exploit a known baseline of a client, allowing an adversary to reconstruct absolute private inputs.



	Concept	Leakage
Zero-Knowledge-Proofs	**Prover** convinces **Verifier** a statement is true without revealing *any details* of the proof	**Homogeneity attack:** if all k records share a sensitive value, it is directly revealed
k-**Anonymity**	Each record is indistinguishable from $\geq k-1$ others on quasi-identifiers	**Homogeneity attack:** if all k records share a sensitive value, it is directly revealed
Trusted Execution Environment (TEE)	Code & data run in a hardware-isolated enclave (e.g. Intel SGX)	**Hardware side-channels:** Spectre, Meltdown, SGX cache-timing attacks

	Concept	Leakage
Synthetic Data Generation	ML model generates artificial data that mirrors real-data statistics	**Membership inference:** model may memorise individual training records
Secret Sharing (e.g. Shamir)	Secret split into n shares; only $\geq t$ shares allow reconstruction	**Threshold compromise:** t colluding parties can fully reconstruct the secret
Blind Signatures	A signer creates a valid signature on a message without ever seeing its content.	**Unlinkability failures:** can leak linkage between the signing session and the final signature if the scheme is poorly implemented or the signer is malicious.

Now that we know

how **PETs work**,

and that they **DO leak**,

we can try to use that leakage to

break them

PET Attacks

Lets play a


Metadata
Traffic Analysis
Inference
Side Channels
Statistical Leakage
Game of Leaks
Information is coming.
And nothing stays secret.

In all of the following attacks you are the

honest-but-curious  provider

- Attack A: (General) Metadata Leakage
- Attack B: Differential Privacy
- Attack C: Federated Learning
- Attack D: Homomorphic Encryption
- Attack E: Multiparty-Computation

☠ Attack A:
(General) Metadata Leakage

Attack A: Metadata Group Communication Leakage

You cannot see:

- Messages
- Files
- Queries
- Encrypted Payloads

You can only see:

- Sender
- Receiver
- Timestamp
- CT Size

Question:
Can you still learn something useful?

Attack A: Your Mission

You receive:

- Network logs
- Message timestamps
- Communication partners

Rule:
You may never inspect message contents.

Logs:
`https://aisec.bramm.xyz/a`

You should find:

1. Who is the CEO?
2. Who is the manager / communication hub?
3. Which users form the developer team?
4. Which users belong to security?
5. Which users work at night?
6. When did an incident happen?
7. Who is the highest-value target?

Attack A: Your Mission

You receive:

- Network logs
- Message timestamps
- Communication partners

Rule:
You may never inspect message contents.

Logs:
`https://aisec.bramm.xyz/a`

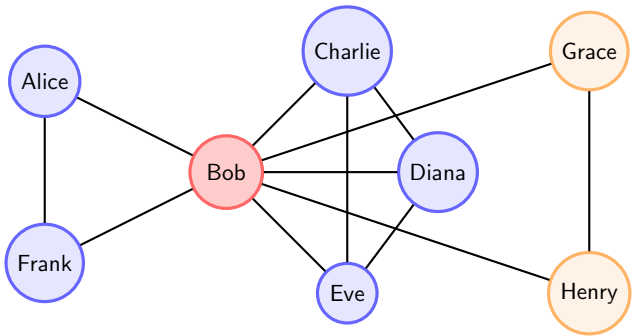
Hint:
`https://aisec.bramm.xyz/a/hint`

You should find:

1. Who is the CEO?
2. Who is the manager / communication hub?
3. Which users form the developer team?
4. Which users belong to security?
5. Which users work at night?
6. When did an incident happen?
7. Who is the highest-value target?

Step 1: Reconstructing the Unweighted Social Graph

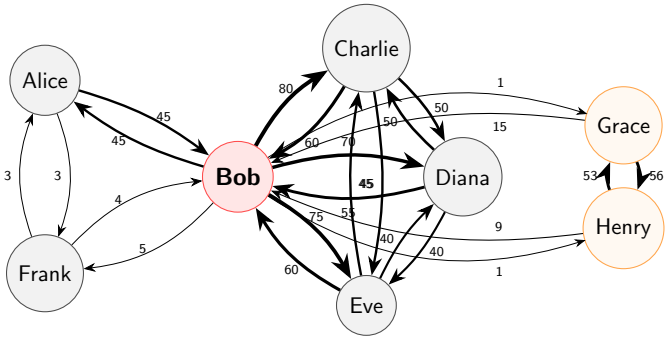
First, we map out the connections (who talks to whom) to understand network topology.



Observation: A clear **Star-Mesh Hybrid** emerges around **Bob** (7 total contacts), while the night shift (**Grace & Henry**) maintains a tight sub-cluster with minimal management crossover.

Step 2: Weighting the Traffic Channels

Adding directional message volume reveals actual operational dependencies.



Thick lines indicate heavy operational pipelines (> 50 messages).

Step 3: Temporal Analysis & Incident Detection

Correlating traffic volume with time zones reveals behavioral shifts and anomalies.

The Night Shift Group

While the core network sleeps, an explicit anomaly appears between two specific nodes:

- **Grace:** 71 night messages
- **Henry:** 62 night messages
- **Bob:** 11 night messages (Minimal overlapping oversight)

Traffic Spike Anomaly (The Incident)

Timestamp (UTC)	Volume	Classification
2026-06-04 **03:00:00**	**35 messages**	**CRITICAL ANOMALY**
2026-06-04 08:00:00	23 messages	Standard Morning Peak
2026-06-05 15:00:00	18 messages	Baseline Operations

Definitive Solutions: Roles & Teams (Q1 - Q4)

Q1: The CEO **Alice**

Deduction: Low total contact footprint, but high vertical interaction with Bob (45). This is the standard fingerprint of an executive shielded by a chief of staff or core manager.

Q2: The Manager **Bob**

Deduction: The absolute communication hub. Highest total network contacts (7) and dominant aggregate message volume.

Q3: Developer Team **Charlie, Eve, Diana, and Frank**

Deduction: They form the active operational cluster. Charlie, Eve, and Diana form the tight inner dev-clique, while **Frank** acts as an integrated engineer interacting directly with both management (Bob) and administration (Alice).

Q4: Security Team **Grace and Henry**

Deduction: Highly isolated pair with continuous operational telemetry exclusively between one another.

Definitive Solutions: Events & Vectors (Q5 - Q7)

Q5: Night Workers **Grace and Henry**

Deduction: Explicitly isolated via the temporal traffic profiles (71 and 62 nocturnal packets).

Q6: Incident Time **2026-06-04 at 03:00:00 UTC**

Deduction: A massive outlier spike (35 messages) hitting right during the security night shift window.

Q7: Highest-Value Target **Alice (Strategic) or Bob (Operational)**

Deduction:

- Compromising **Alice** yields executive keys, legal authorization, and macro data.
- Compromising **Bob** completely map-exposes the ongoing active engineering pipeline.

The encryption worked !

But the metadata leaked:

- Social graphs
- Organizational structure
- Behavioral patterns
- High-value targets

Bonus: We didn't even touch CT size in our analysis

Lesson:

Protecting content is not the same as protecting privacy.

☠ Attack B: Differential Privacy

Attack B: Differential Privacy

Database:

Patients with a rare cancer disease

Access is protected by Differential Privacy.

Every answer contains random noise.

Question:
How can we circumvent/break DP?

Attack B: Your Mission

You receive:

- Access to DPed cancer database
- API: `https://aisec.bramm.xyz/b/?`

Rule:
Use console scripts, curl (and jq if you want to) only!

You may repeatedly ask:

- How many patients have cancer?
- How many patients are in which age cohort?
- Your job: Infer all hidden information

Attack B: Your Mission

You receive:

- Access to DPed cancer database
- API: `https://aisec.bramm.xyz/b/?`

Rule:
Use console scripts, curl (and jq if you want to) only!

Hint:
`https://aisec.bramm.xyz/b/hint`

You may repeatedly ask:

- How many patients have cancer?
- How many patients are in which age cohort?
- Your job: Infer all hidden information

Overview: Two Fundamental Attack Classes

- **Attack B.1: Noise Cancellation (Repeated Queries)**
- **Attack B.2: Structure Reconstruction (Query Sweeps)**

Attack B.1: Noise Cancellation

Goal: Recover a hidden value despite DP noise.

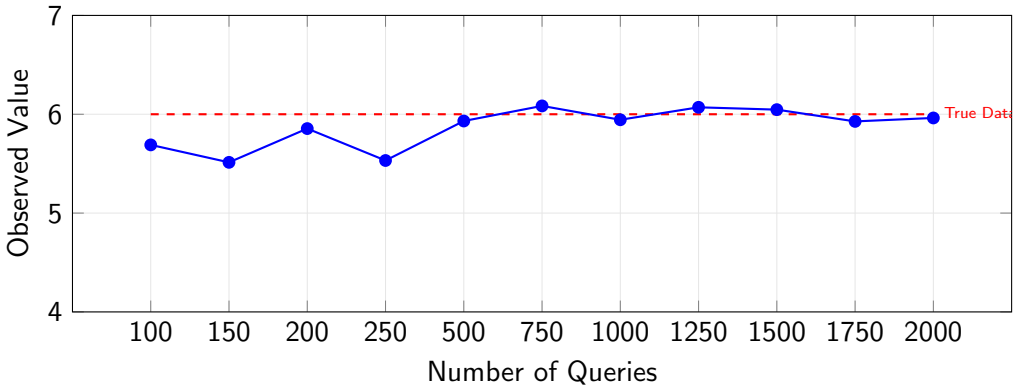
We query the same function multiple times:

$$f(D) + \mathcal{N}(0, b)$$

Observation:

- Noise is zero-mean
- Averaging reduces variance
- Signal emerges over repeated queries

Attack B.1: Averaging Effect



Result: Mean converges to real value = 6

Attack B.2: Structure Reconstruction (Query Sweeps)

Even with Differential Privacy noise ($\sigma = 2.0$), sliding a threshold filter leaks structural boundaries.

The Core Vulnerability

Every step-increment in the query range modifies the target subpopulation by a deterministic amount:

$$\text{True Count}(\text{Age} > t) - \text{True Count}(\text{Age} > t + \Delta) = \text{Count}(t < \text{Age} \leq t + \Delta)$$

Why Independent Noise Fails Here

If we look at the raw data vector returned by the loop, the numbers reveal exact demographic clusters wherever a sudden drop-off overrides the baseline background noise.

Attack B.2: Structure Reconstruction (Query Sweeps)

Here is why this makes the attack so devastatingly (in simple terms):

The Server Sweeps Itself

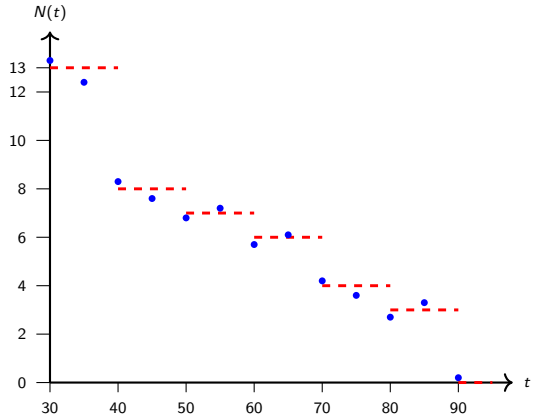
When you visit `/?reconstruct`, the backend code instantly runs a for loop that tests every age range (range(30, 90, 5)). It collects all the answers, bundles them into a single package, and hands them back to you in one go.

The Noise Doesn't Sync Up

Even though the server injects fresh random noise into each step of the loop, it does all of them independently. Because you get the entire list of noisy results at once, you can immediately look at the drops between neighboring age groups.

Step 2: Denoising the Cumulative Sweep Step Function

Averaging results reveals the underlying step changes where real patients exist.



Sharp drops between intervals point directly to the presence of specific user records.

Step 3: Reconstructed Bin Analysis

Subtracting adjacent outputs filters out the noise floor and maps the target demographic layout.

Calculated Deltas ($\Delta N = N(t) - N(t+5)$)

Age Window	Calculated Mean Delta	Reconstructed Population
30 – 40	$\approx$ 5.0 items	High Density Cluster
40 – 50	$\approx$ 1.0 items	Single Target Present
50 – 60	$\approx$ 1.0 items	Single Target Present
60 – 70	$\approx$ 2.0 items	Duplicated Cluster
70 – 80	$\approx$ 1.0 items	Single Target Present
80 – 90	$\approx$ 3.0 items	High Age Cohort

This step completely bypasses the protections of individual route query limits.

The real values are:

```
$PATIENTS = [
    ["age" => 31, "cancer" => 0],
    ["age" => 36, "cancer" => 0],
    ["age" => 33, "cancer" => 0],
    ["age" => 34, "cancer" => 0],
    ["age" => 35, "cancer" => 0], // 30-40: 5 people
    ["age" => 45, "cancer" => 0], // 40-50: 1 people
    ["age" => 52, "cancer" => 0], // 50-60: 1 people
    ["age" => 61, "cancer" => 1],
    ["age" => 67, "cancer" => 1], // 60-70: 2 people
    ["age" => 78, "cancer" => 1], // 70-80: 1 people
    ["age" => 81, "cancer" => 1],
    ["age" => 83, "cancer" => 1],
    ["age" => 86, "cancer" => 1], // 80-90: 3 people
];
```

Attack B: Challenge Solutions

Q1: Is the database safe? **No.** The privacy budget (ϵ) is exhausted almost instantly because individual queries are linked across a logical continuum.

Q2: What is leaked? **The exact age topology** and **the exact number of cancer patients.** An adversary can identify missing age ranges, detect high-risk old-age individuals, and isolate single counts.

Q3: Mitigation Strategy **Implement a Global Privacy Budget / Composition Check.**

- Add a central mechanism that tracks total query budget consumed.
- Inject correlated noise across related threshold parameters instead of independent noise.

Attack B: What Leaked?

Differential Privacy protects:

- Individual contributions

From leakage attacks we can infer:

- Population distribution
- Clusters / cohorts
- Sensitive subgroups
- Structural anomalies
- number of cancer patients

Lesson:

Global privacy budgets matter.

Attack C: Federated Learning

Attack C: Federated Learning

Raw training data never leaves the device.

Only gradients are shared.

Can gradients reveal the original training sample?

Attack C: Your Mission

You receive:

- Weighted (binary) gradients
- Attack Script

Goal, reconstruct:

- Model parameters
- Input features

using gradient inversion.

Rule:
Recreate the model (*model.py*). Then use *attack.py* to reconstruct input features by gradient inversion of *input.pt*.

Files:
`https://aisec.bramm.xyz/c.zip`

Attack C: Your Mission

You receive:

- Weighted (binary) gradients
- Attack Script

Goal, reconstruct:

- Model parameters
- Input features

using gradient inversion.

Rule:
Recreate the model (*model.py*). Then use *attack.py* to reconstruct input features by gradient inversion of *input.pt*.

Files:
`https://aisec.bramm.xyz/c.zip`
Hint:
`https://aisec.bramm.xyz/c/hint`

The Federated Privacy Fallacy

Federated Learning claims to protect privacy by keeping data localized and only exchanging model parameters or updates (∇W).

The Gradient Core Relation

Gradients are calculated using the chain rule during backpropagation:

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \mathbf{x}$$

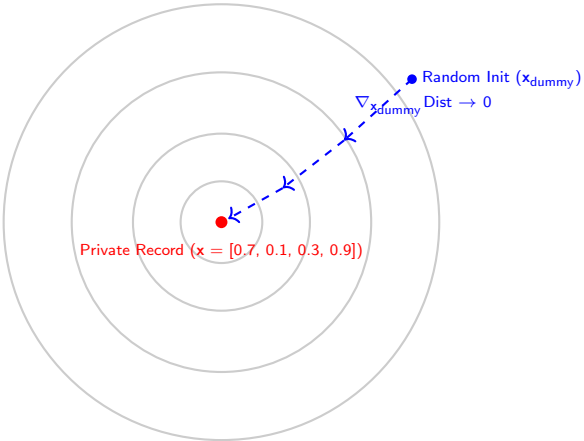
Because the weight architecture W and the output loss function are known public entities, the gradient remains a strict linear scaling of the private input vector $\mathbf{x}$.

Deep Leakage Mechanics

An adversary intercepting ∇W can turn a standard machine learning model inside out, treating the input features as an optimization problem to extract original training rows.

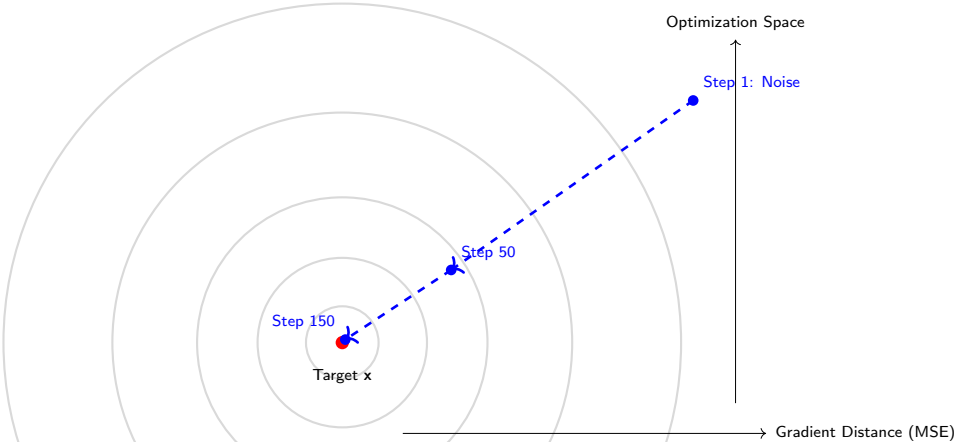
Inversion Optimization Space

The attack iteratively warps random noise ($\mathbf{x}_{\text{dummy}}$) into the patient record by minimizing the Mean Squared Error between matching backpropagation steps.



  Fraunhofer AISEC

Reconstruction Evaluation



Convergence Analysis:
Step 1: Random Gaussian distribution.
Step 50: Structural features recognizable.
Step 150: MSE = 0.00000000. Data fully inverted.

Step 1: Inspecting the Raw Binary Matrix Dimensions

Without `model.py`, running the attacker script fails immediately. You must act as a reverse-engineer and read the tensor metadata from the binary file using a simple Python interactive command string on the terminal:

CLI Metadata Inspection Command

```
python -c "import torch; b = torch.load('leak.pt'); print({k: v.shape for k, v in b['state'].items()})"
```

Resulting Terminal Output

```
{
  'net.0.weight': torch.Size([16, 4]),
  'net.0.bias': torch.Size([16]),
  'net.2.weight': torch.Size([2, 16]),
  'net.2.bias': torch.Size([2])
}
```

Step 2: Mapping Tensors to Structural Layers

By analyzing the array geometry of the leaked state dictionary keys, you map out the network configuration matrices linearly:

`net.0.weight` → Size([16, 4])]
: The transformation matrix matches the dimension format Size([Outputs, Inputs]). This confirms the first layer takes an input vector of length **4** and maps it to a hidden space of **16** neurons.

`net.1`
: Missing from the state dictionary keys. Because it holds no weight matrices or bias vectors, it must be a parameterless non-linear function like **ReLU**.

`net.2.weight` → Size([2, 16])]
: The final transformation layer takes the **16** internal dimensions and reduces them down to an output array of length **2** (the logits for binary classification).

Step 3: Reconstructing `model.py`

The Recovered Blueprint Code (`model.py`)

```python
import torch.nn as nn

class SmallNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(4, 16),    # Deduced from weight shape [16, 4]
            nn.ReLU(),           # Implied non-linear bridge layer
            nn.Linear(16, 2)     # Deduced from weight shape [2, 16]
        )

    def forward(self, x):
        return self.net(x)
```

Recreating this SmallNet[1] unblocks the namespace imports in `attacker.py`, allowing the optimization loop to successfully execute and recover the values.

[1] usually refers to a compact, lightweight Convolutional Neural Network (CNN) specifically designed for basic image classification tasks like MNIST digit recognition.

Reconstruction Evaluation

Using a single gradient vector update exported to `leak.pt`, the L-BFGS[2] routine strips away the parameter variance completely within 150 steps.

Dimension Block	Original Baseline Data	Reconstructed Vector
Feature x_0	0.7000	0.7000
Feature x_1	0.1000	0.1000
Feature x_2	0.3000	0.3000
Feature x_3	0.9000	0.9000
Target Label y	**Class 1**	**Class 1 (100% Match)**

Inversion Resolution

The tracking loss drops to absolute zero (0.00000000), demonstrating that gradient updates preserve spatial properties perfectly without noise protection.

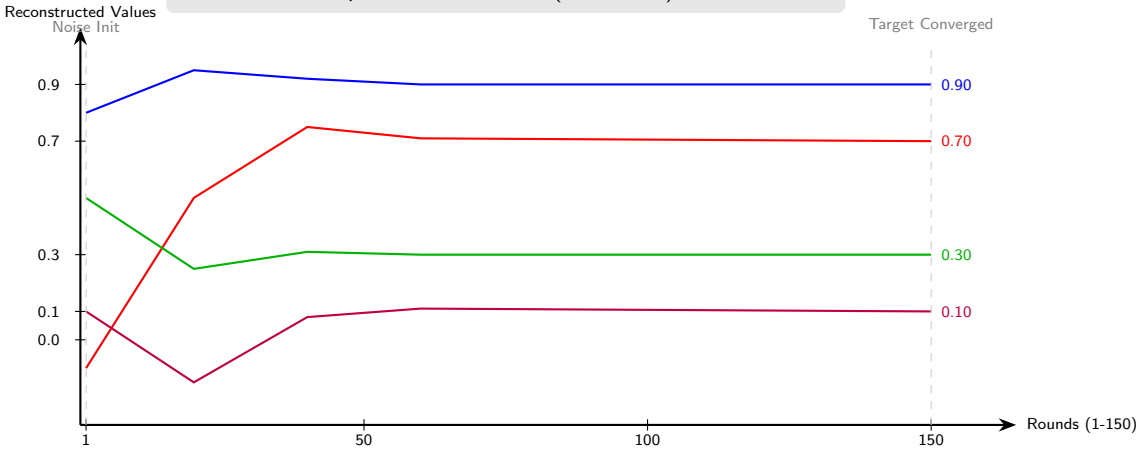
[2] Limited memory-Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS)

Reconstruction Evaluation

Convergence Key:

Distance $= \sum \left\| \nabla W_{\text{victim}} - \nabla W_{\text{dummy}} \right\|^2$

The distance collapses to absolute zero (MSE=0.00) in <150 rounds.



Defensive Engineering: Gradient Sanitization

To participate safely in distributed parameter aggregation, networks must decouple raw updates from deterministic recovery.

1. **Differential Privacy (DP-SGD):** Add controlled Gaussian or Laplacian noise directly to the computed gradients before serialization. Blur the feature fingerprint while retaining statistical learning value.
2. **Secure Multi-Party Computation (SMPC):** Encrypt individual updates such that the central aggregator can only decode the combined, averaged global model parameter step $\sum \nabla W_i$, keeping individual nodes blind.
3. **Gradient Clipping:** Bound maximum updating thresholds to restrict structural scaling dimensions.

☠ Attack D:
Homomorphic Encryption

Attack D: Homomorphic Encryption

The cloud can compute on encrypted data.

The cloud never sees plaintext.

Can an attacker still infer sensitive information?

Attack D: Your Mission

You receive:

- Ciphertext size
- Request timing
- Processing time

Goal, infer:

- User behavior
- Query type
- Disease category

without decrypting anything.

Rule:
This time you should inspect the python source and adopt it to attack the client.

Files:
`https://aisec.bramm.xyz/d.zip`

Attack D: Your Mission

You receive:

- Ciphertext size
- Request timing
- Processing time

Goal, infer:

- User behavior
- Query type
- Disease category

without decrypting anything.

Rule:
This time you should inspect the python source and adopt it to attack the client.

Files:
`https://aisec.bramm.xyz/d.zip`

Hint:
`https://aisec.bramm.xyz/d/hint/attack.py`

The Illusion of Total Privacy

Homomorphic Encryption (HE) allows computation on encrypted data, but it does not automatically hide the **structure** of the data.

The "Shape" Leak

- **Encrypted Values:** Protected. The server sees random noise.
- **Data Dimensions:** Leaked. The size of the ciphertext (Size_{CT}) is a function of the input vector length (L).

The Side-Channel

In the provided code, the server processes data at a rate of $50,000$ bytes/sec. This turns **Size** into **Time**, creating two ways to identify the disease.

The Side-Channel Mechanism: Background Profiling

An observer watching an isolated ciphertext stream natively sees only randomized binary data. The semantic mapping relies on a two-phase traffic analysis model:

1. Domain Context (Targeting)

   The adversary does not guess blindly in a vacuum. By evaluating network-layer metadata (*IP addresses, DNS queries, or SNI headers*), they isolate the target endpoint:

   $$\text{Destination} = \texttt{telehealth-portal.de/infer} \implies \mathcal{M} = \{\text{Medical Diagnoses}\}$$

   This eliminates unrelated domains like travel routes or automotive data entirely.

2. Offline Profiling (Building $\mathcal{A}_{\text{lookup}}$)

The attacker registers a legitimate account or hosts a local instance of the target framework. By executing known control inputs, they map the deterministic cryptographic choices of the system:

$$f(\text{“Flu”}) \longrightarrow \mathbf{c}_{\text{flu}} \implies S \approx 80.9\,\text{KB}$$
$$f(\text{“Cancer”}) \longrightarrow \mathbf{c}_{\text{cancer}} \implies S \approx 1.05\,\text{MB}$$

Because the developer dynamically scaled the polynomial modulus degree (N) and prime bit-depth to fit each disease vector, they baked a permanent structural fingerprint into the ciphertext length.

3. Online Passive Interception

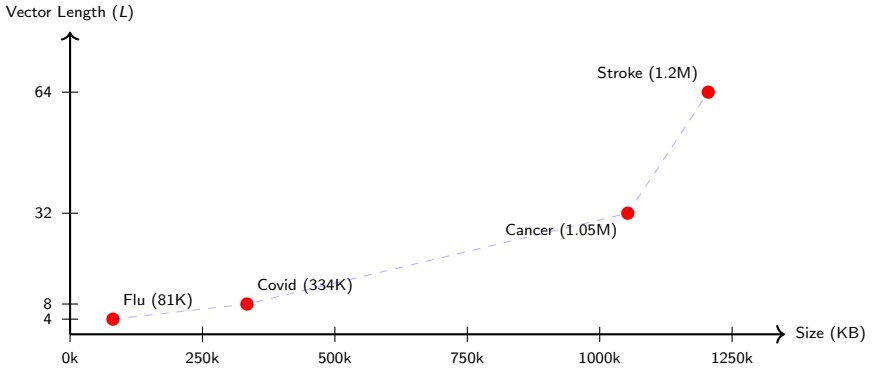
During live execution, the attacker sits passively on the wire. When an anonymous patient transmits a blind ciphertext $\mathbf{c}_{\text{target}}$:

$$\text{Read Size}(\mathbf{c}_{\text{target}}) = 1,053,100 \text{ B} \xrightarrow{\mathcal{A}_{\text{lookup}}} \text{CANCER}$$

The data values remain mathematically hidden inside the ciphertext, but the **cryptographic parameter wrapper** completely breaks the patient's privacy model.

Step 1: Mapping Parameter Step Leakage

By matching crypto-parameters to disease severity, we created a "size signature" that spans three orders of magnitude.



Result: A 15× size difference between Flu and Stroke makes de-anonymization trivial for any network observer.

Step 2: The Profiling Matrix

An adversary creates an empirical lookup table offline. When a patient transmits an inference payload, simple dictionary matching breaks the security model.

Observed Size	Server Latency	Dimension	Inferred State
$\approx 80,900$ B	≈ 7 ms	4	**Flu**
$\approx 334,400$ B	≈ 19 ms	8	**Covid**
$\approx 1,053,100$ B	≈ 72 ms	32	**Cancer**
$\approx 1,204,800$ B	≈ 66 ms	64	**Stroke**

The Architectural Trap

The system designer successfully hid the **data values** inside the ciphertext, but by varying the **cryptographic parameters** (Degree & Primes) to fit the disease, they created an unmistakable digital fingerprint.

Note: Note that Stroke is slightly faster than Cancer despite being larger, due to differing bit-modulus complexity.

Exploit Execution Pipeline

The attack cycle relies distinct process environments:

1. **Honest-but-curious Server (`server.py`):** The target node boots the inference engine on port 5001.
2. **The Data Leak Capture (`client.py`):** The user context transmits encrypted arrays to port 5001.
3. **The honest-but-curious Server learns the** ATTACKER_LOOKUP table based on logged queries.
4. **De-anonymization Mapping (`attacker.py`):** The profiling framework maps the active packet size to the target dictionary.

Defensive Mitigations

To eliminate metadata side-channels, we must transform structural components into invariants.

Vector Padding Force all incoming plaintext arrays to a uniform static dimension (e.g., $L = 64$). Fill unused elements with zero-padding prior to processing the polynomial transformations.

Constant-Time Execution Decouple runtime computing parameters from payload length metrics. Force the application backend to evaluate and output at a static upper-bound timing window.

Fixed-Length Serialization Pad outgoing binary transport buffers to ensure that $\text{Size}(\text{Flu}) \equiv \text{Size}(\text{Stroke})$.

Attack D: What Leaked?

Homomorphic Encryption protected:

- Plaintext values
- Computation

The system leaked:

- Metadata
- Access patterns
- Timing information

Lesson:

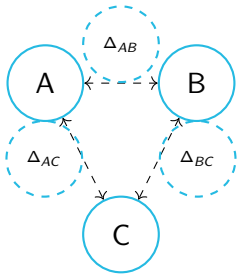
The cryptography worked.
The system failed.

Attack E:
Multi-Party-Computation

The Multi-Clinic Scenario

In a research network, 3 clinics perform pairwise similarity checks to find matching patient cohorts.

- Clinics A, B, and C share secret-shares.
- The Server honestly computes $|V_i - V_j|$ on secret shares.
- The cloud never sees a whole number—only random-looking algebraic noise.
- Cryptography protects the **Plaintext**.



If the math is blind, can a passive server still steal the database?

Attack E: Your Mission

You receive:

- Pairwise secret shares (*shares_generator.sh*)
- Internal loop iterations (I)
- Public baseline data (Clinic C)

Goal, infer:

- Private Clinic A values
- Private Clinic B values

without breaking the secret-sharing encryption.

Rule:
Inspect the loop convergence logic. Solve the linear system by anchoring the graph to a public reference node. Modify the *server.py* to steal the values.

Files:
`https://aisec.bramm.xyz/e.zip`

Attack E: Your Mission

You receive:

- Pairwise secret shares (*shares_generator.sh*)
- Internal loop iterations (I)
- Public baseline data (Clinic C)

Goal, infer:

- Private Clinic A values
- Private Clinic B values

without breaking the secret-sharing encryption.

Rule:

Inspect the loop convergence logic. Solve the linear system by anchoring the graph to a public reference node. Modify the *server.py* to steal the values.

Files:

`https://aisec.bramm.xyz/e.zip`

Hint:

`https://aisec.bramm.xyz/e/hint/attack.py`

The "Metadata Convergence" Trap

Secure Multi-Party Computation (SMPC) protects data values, but it often leaks the **structural distance** through execution metadata.

The Logic Leak

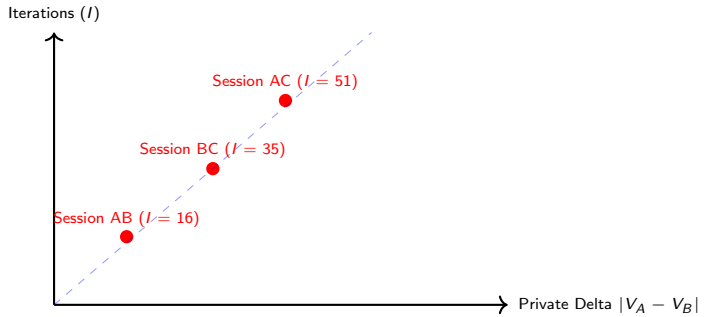
- **Encrypted Shares:** Secure. The server sees random field elements.
- **Loop Iterations (l):** Leaked. The protocol runs until $|V_i - V_j|$ converges.

The Side-Channel

The number of iterations l is a direct function of the absolute difference between two private values. This turns a "blind" calculation into a **relative distance map**.

Step 1: Mapping the Relative Network

The Honest-but-Curious server logs I for every pairwise clinic comparison. It constructs a system of equations without decrypting a single share:



Result: The server now knows the *distance* between all clinics, but the absolute values are still "floating" in space.

Step 2: The Anchor Breakthrough

The adversary identifies an **Anchor Clinic**—a node whose value is public or a known baseline (e.g., a Public Control Group: Clinic C).

Linear Reconstruction

Once the floating graph is pinned to a single known point, the whole system collapses:

$$\text{Public Anchor (Clinic C)} = \mathbf{10}$$

$$\text{Inferred Clinic B} = V_C + \Delta_{BC} = 10 + 35 = \mathbf{45}$$

$$\text{Inferred Clinic A} = V_B + \Delta_{AB} = 45 + 16 = \mathbf{61}$$

The server learned the exact database without ever reading the shares.

Defensive Mitigations

To stop an Honest-but-Curious server, we must decouple data from execution time.

Fixed-Loop Padding Force every SMPC comparison to run for exactly I_{max} iterations. If the result is found early, the server must perform "dummy" math to hide the exit point.

Differential Privacy Add noise to the loop termination signal so that I is only a probabilistic approximation of the distance.

Blinded Routing Use Onion Routing so the server does not know which clinic labels (A, B, C) correspond to which encrypted payloads.

Attack E: What Leaked?

SMPC protected:

- Private plaintext values
- Intermediate algebraic states

The system leaked:

- Convergence metadata
- Pairwise structural deltas
- Global network topology

Lesson:

A server that is "Honest" enough to run the code, but is "Curious" enough to count the steps.

🏁 Conclusion

Ask better questions.

Shift the mindset. That's where the real work begins.

	Wrong mindset	Better mindset
"Is it secure/private?"	binary	leakage spectrum
"Is HE secure?"	theoretical	operational
"Can PETs solve privacy?"	magical	contextual

Ask better questions.

Shift your mindset to:

Leakage Surface Thinking

Conclusion: What did we do?

	Abused Leakage	Inferred Truth
A General Metadata	Traffic Analysis / Times	Social Graph, Targets
B Differential Privacy	Statistical Leakage	Exact Plaintext
C Federated Learning	Gradients	Exact Plaintext
D Homomorphic Encryption	CT Size / Latency	Disease Category
E Multiparty-Computation	Convergence Loops	Exact Plaintext

The Final Lesson

The cryptography worked perfectly in every attack.
The **system architecture** failed by allowing structural metadata to leak the underlying state.

References

- Cynthia Dwork and Aaron Roth, *The algorithmic foundations of differential privacy*, Foundations and trends® in theoretical computer science **9** (2014), no. 3-4, 211–487.
- Craig Gentry, *Fully homomorphic encryption using ideal lattices*, Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009, pp. 169–178.
- Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas, *Communication-efficient learning of deep networks from decentralized data*, Artificial intelligence and statistics, Pmlr, 2017, pp. 1273–1282.
- Andrew C Yao, *Protocols for secure computations*, 23rd annual symposium on foundations of computer science (sfcs 1982), IEEE, 1982, pp. 160–164.

Thanks for listening

Fraunhofer
AISEC

Contact

Department
Applied Privacy Technologies
Tel. +49 89 3229986-147
Fax +49 89 3229986-222
georg.bramm@aisec.fraunhofer.de

Fraunhofer-Institute for Applied and Integrated Security AISEC
Lichtenbergstr. 11
85748 Garching (near Munich)
Germany