Lecture #7

Advanced Mobile Privacy

SMA 2025

Advanced Mobile Privacy: From Permissions to Formal Models

Beyond "Allow" and "Deny"

Today's Agenda

- **Part 1: The Data Gold Rush** - The mobile privacy landscape.
- **Part 2: The Gatekeepers Revisited** - A deep dive into permission systems.
- **Part 3: The Privacy Toolkit** - An introduction to Privacy-Enhancing Technologies (PETs).
- **Part 4: Hiding in the Crowd** - The formal model of k-Anonymity.
- **Part 5: Statistical Secrecy** - The formal model of Differential Privacy.
- **Part 6: The Road Ahead** - Summary and what's next.

Recap from Lecture 6

- **Identity & Access Management (IAM):** We explored how we prove our identity on mobile.
- **Authentication Factors:** Something you know, have, and are.
- **Federated Identity:** We learned how OAuth 2.0 and OpenID Connect (OIDC) with PKCE enable secure "Sign in with..." flows.
- **The Future is Passwordless:** We saw how Passkeys are set to replace passwords.

Today's Link: Once a service knows who you are, privacy controls determine what they are allowed to know about you.

Part 1: The Data Gold Rush

The Mobile Privacy Landscape

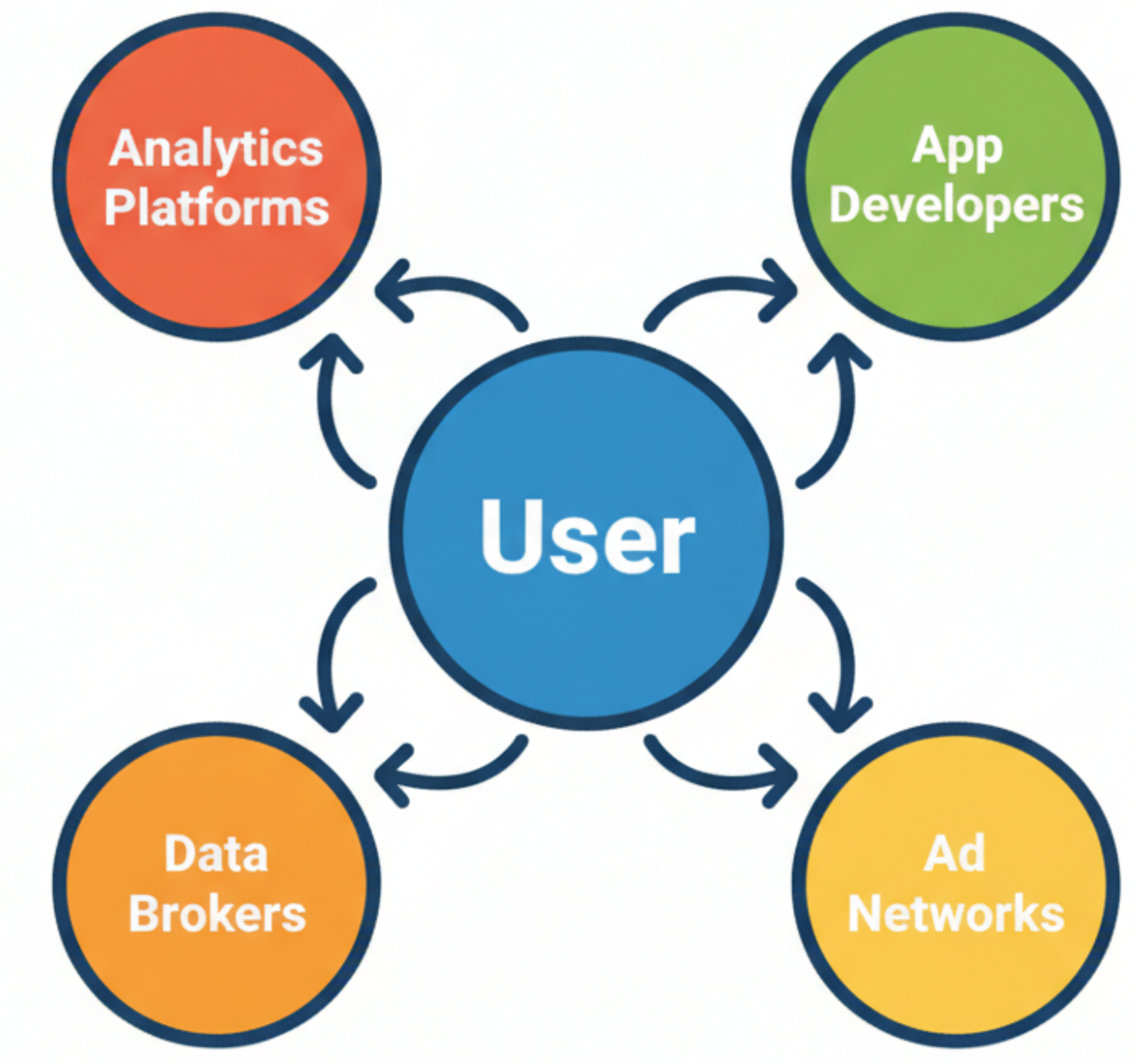
Why Your Data is Valuable

Personal data is the fuel for the modern digital economy, primarily for:

- **Targeted Advertising:** Showing you ads you are more likely to click on.
- **Personalization:** Customizing app experiences to your preferences.
- **Analytics & Insights:** Helping businesses understand their customers.
- **Risk Assessment:** Used by financial institutions for credit scoring and fraud detection.

The Players in the Ecosystem

- **App Developers:** They collect data to improve their app and to monetize it.
- **Ad Networks (e.g., Google AdMob, Meta Audience Network):** They provide the tools (SDKs) for developers to show ads.
- **Analytics Platforms (e.g., Firebase, Mixpanel):** They provide tools to track user behavior inside the app.
- **Data Brokers (e.g., Acxiom, Experian):** The hidden players. They buy data from various sources, aggregate it, and sell detailed user profiles.



What is a "Tracker"?

A tracker is any piece of code included in an app for the purpose of collecting data about you or your device for third-party use.

- **The Mechanism:** Usually delivered via a third-party SDK.
- **Example:** A game developer includes an Ad Network SDK. That SDK is a **tracker**. It might collect:
  - Your device's unique Advertising ID.
  - Your location.
  - Your device model, OS version, and language.
  - A list of other apps installed on your device.

Trackers in Your Code (Android)

```
// app/build.gradle.kts

dependencies {
    // ... other dependencies
    implementation("com.google.firebase:firebase-analytics:21.5.0")
    implementation("com.meta.android-sdk:facebook-android-sdk:latest.release")
    implementation("com.mixpanel:mixpanel-android:7.3.2")
}
```

The Advertising ID

The key that connects all the dots.

- **What it is:** A unique, user-resettable ID for advertising on a device.
  - **IDFA** (Identifier for Advertisers) on iOS.
  - **AAID** (Android Advertising ID) on Android.

Accessing the Ad ID in Code (Android)

This shows how an app would get the Android Advertising ID (AAID). This now requires the *AD_ID* permission.

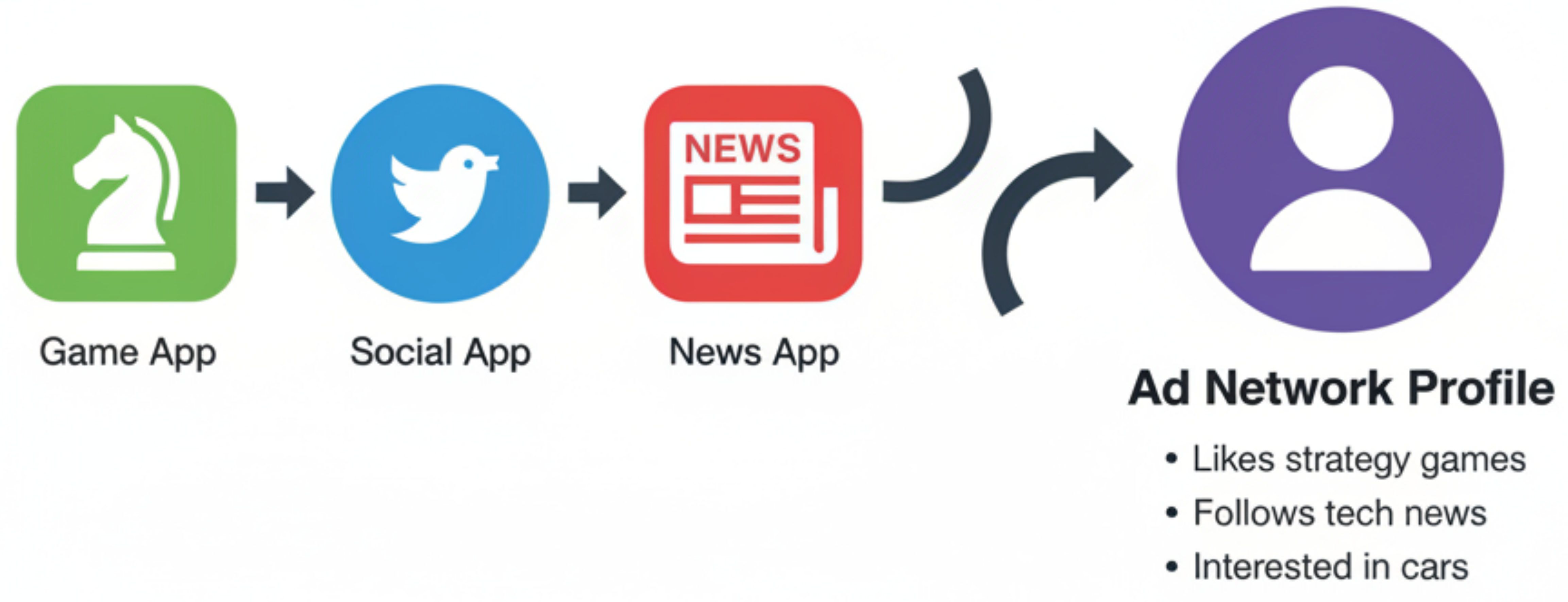
```
// This requires the com.google.android.gms.permission.AD_ID permission in the manifest

import com.google.android.gms.ads.identifier.AdvertisingIdClient

// This must be called on a background thread
fun getAndroidAdvertisingId(context: Context): String? {
    return try {
        val adInfo = AdvertisingIdClient.getAdvertisingIdInfo(context)
        adInfo.id
    } catch (e: Exception) {
        // User has opted out of ad tracking, or Google Play Services are not available.
        null
    }
}
```

Accessing the Ad ID in Code (iOS)

```swift
import AdSupport
import AppTrackingTransparency

func requestAdvertisingId() {
    // Request permission from the user
    ATTrackingManager.requestTrackingAuthorization { status in
        DispatchQueue.main.async {
            if status == .authorized {
                // User granted permission. We can now access the IDFA.
                let idfa = ASIdentifierManager.shared().advertisingIdentifier
                print("IDFA: \(idfa.uuidString)")
            } else {
                // User denied permission. The IDFA will be all zeros.
                print("Tracking not authorized.")
            }
        }
    }
}
```

The Privacy Problem: Cross-App Tracking

This is the core privacy concern. By observing your behavior in many different apps via the Advertising ID, the ad network can build a rich, detailed profile of your habits, interests, and personal life.



Beyond the Ad ID: Fingerprinting

What if the user denies access to the Ad ID?

Trackers have a backup plan: **Device Fingerprinting**.

- **Concept:** Create a unique identifier by combining many non-unique pieces of information about a device.
- **Data Points Used:**
    - IP Address, Carrier, Country Code
    - Device Name, Model, Screen Resolution
    - OS Version, Build Number
    - List of installed fonts
    - Language, Timezone

Fingerprinting in Code (Conceptual)

This is how a tracker might gather data points for a device fingerprint.

```
fun createDeviceFingerprint(): String {
    val fingerprint = StringBuilder()
    fingerprint.append(Build.MODEL)
    fingerprint.append(Build.MANUFACTURER)
    fingerprint.append(Build.VERSION.SDK_INT)
    fingerprint.append(Resources.getSystem().displayMetrics.widthPixels)
    fingerprint.append(Resources.getSystem().displayMetrics.heightPixels)
    fingerprint.append(TimeZone.getDefault().id)
    // ... and many more properties

    // Hash the combined string to create a single ID
    return sha256(fingerprint.toString())
}
```

Part 2: The Gatekeepers Revisited

A Deep Dive into Permission Systems

The Goal of a Permission System

To enforce the **Principle of Least Privilege** by giving the user **informed consent** and **granular control** over an app's access to sensitive data and capabilities.

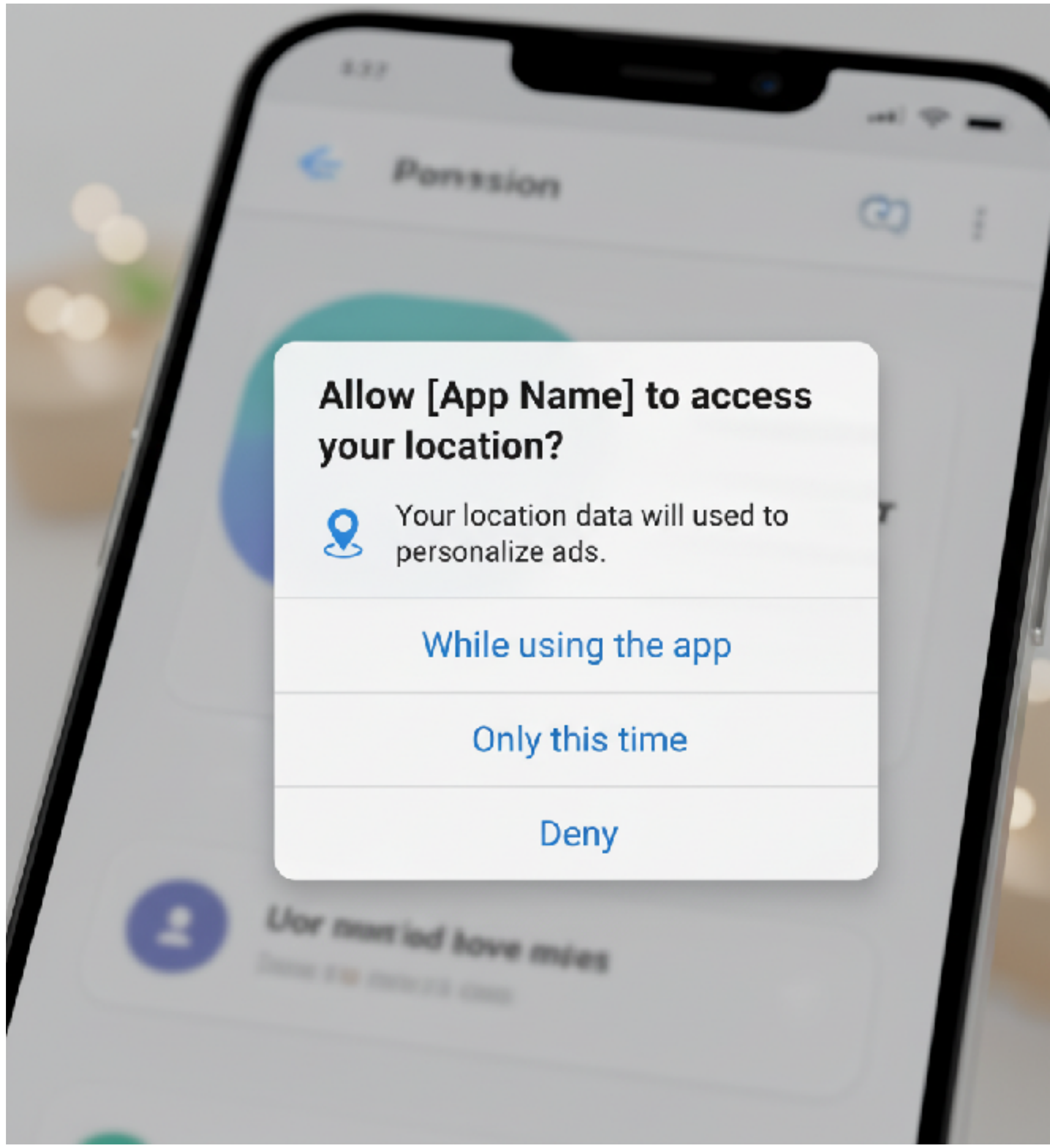
- **Informed Consent:** The user must understand what they are granting access to and why.
- **Granular Control:** The user should be able to grant or deny specific permissions, not just make an "all or nothing" choice.

Android Permissions: A History

- **The Old Way (Pre-Android 6.0): Install-Time Permissions.**You were shown a list of all permissions an app wanted before you installed it.
- Your only choice was to accept everything or not install the app.
- This was "all or nothing" and led to permission fatigue.
- Permissions are requested as they are needed while the app is running.
- Users can grant or deny permissions individually.

Android: One-Time Permissions

- Introduced in Android 11.
- Allows the user to grant a permission for a single session only.
- When the app is closed, the permission is automatically revoked.
- This is great for features you use infrequently, like sharing your location with a ride-sharing app just once.



Android: Auto-Reset Permissions

- Introduced in Android 11.
- If an app is installed but not used for a few months, the system will automatically revoke all the sensitive permissions it was granted.
- The next time the user opens the app, it will have to request them again.
- This prevents old, forgotten apps from sitting on your device and continuing to access your data in the background.

Android Code: Coarse vs. Fine Location

Android allows apps to request either approximate or precise location, giving users more control.

```
// In AndroidManifest.xml
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

// In your Activity
val requestPermissionLauncher = registerForActivityResult(
   ActivityResultContracts.RequestMultiplePermissions()
) { permissions ->
   when {
      permissions.getOrDefault(Manifest.permission.ACCESS_FINE_LOCATION, false) -> {
         // Precise location access granted.
      }
      permissions.getOrDefault(Manifest.permission.ACCESS_COARSE_LOCATION, false) -> {
         // Only approximate location access granted.
      } else -> {
         // No location access granted.
      }
   }
}

// When requesting, you can ask for both. The user can choose to grant only coarse.
requestPermissionLauncher.launch(arrayOf(
   Manifest.permission.ACCESS_FINE_LOCATION,
   Manifest.permission.ACCESS_COARSE_LOCATION))
```

iOS Permissions: Privacy by Design

Apple's approach has always been more stringent and privacy-focused from the start.

- **Runtime by Default:** iOS has always used a runtime permission model.
- **Mandatory Usage Strings:** As we saw in Lecture 2, developers must provide a clear, human-readable reason (*Info.plist* usage string) for why they need a permission. If they don't, the app will crash.
- **No "Allow Always" for Location (Initially):** For a long time, iOS didn't even give users the option to grant permanent, background location access. They later added it, but with prominent reminders.

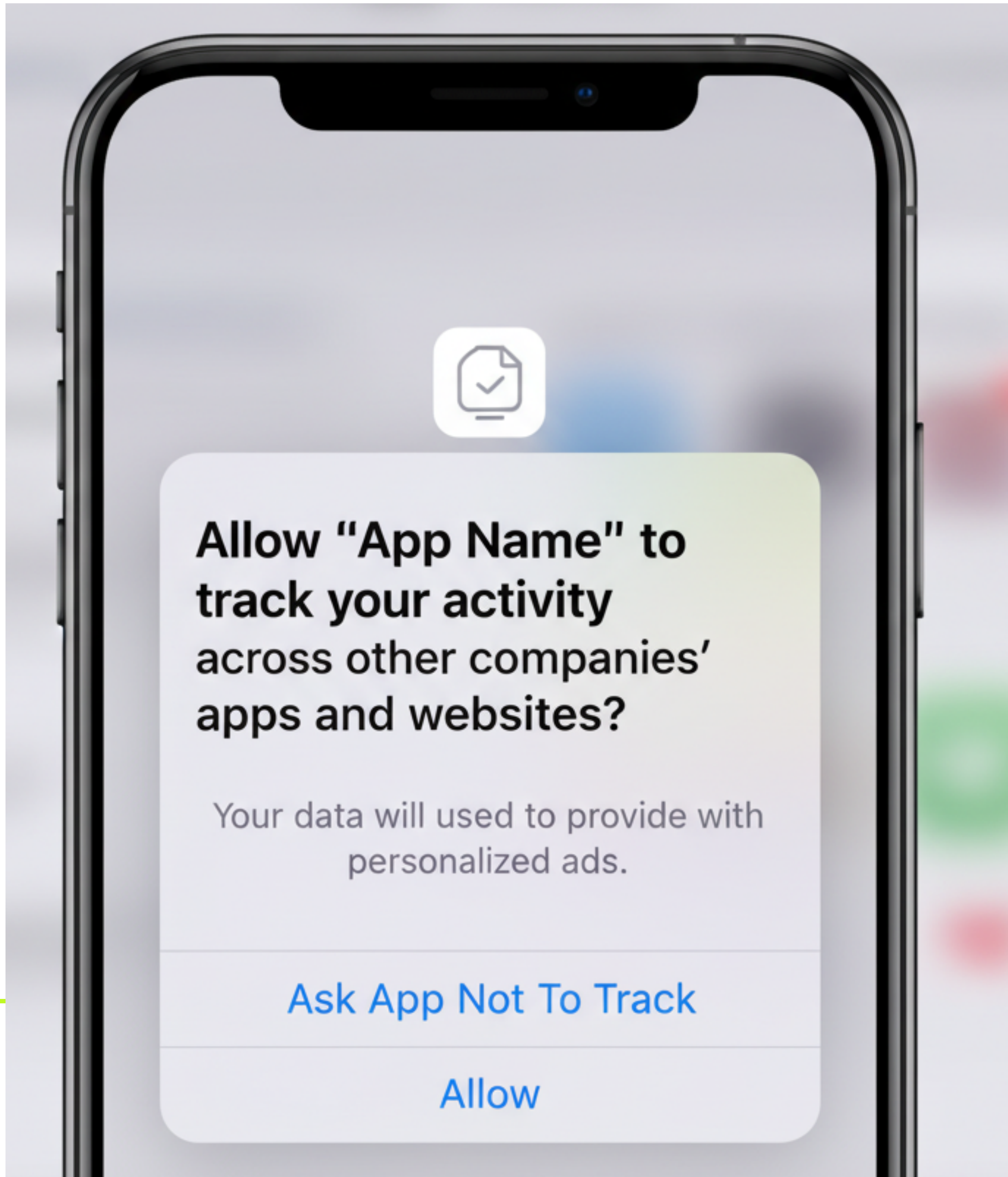
iOS Code: Info.plist Usage Strings

This is what a developer must add to their *Info.plist* file to request permission. The string value is shown directly to the user.

```xml
<!-- Info.plist -->
<key>NSLocationWhenInUseUsageDescription</key>
<string>We need your location to show you nearby restaurants.</string>

<key>NSCameraUsageDescription</key>
<string>We need access to your camera to scan QR codes for payment.</string>

<key>NSContactsUsageDescription</key>
<string>We need to access your contacts to help you find and connect with your friends.</string>

<key>NSMicrophoneUsageDescription</key>
<string>We use the microphone for voice commands.</string>
```

iOS: App Tracking Transparency (ATT)

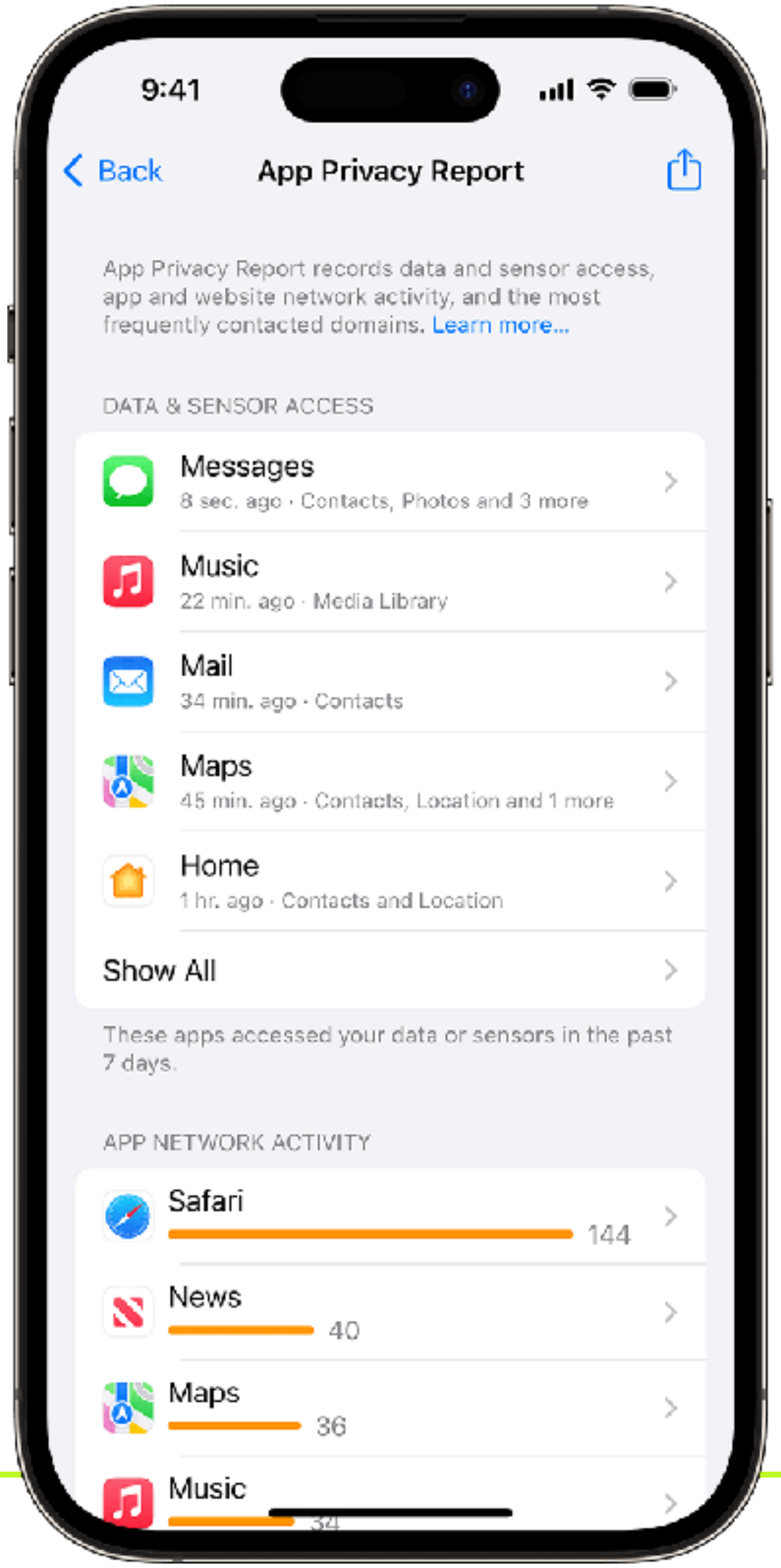
- Introduced in iOS 14.5.
- Requires apps to get explicit user consent before they can access the device's Advertising ID (**IDFA**).
- This single feature has had a massive, disruptive impact on the mobile advertising industry.
- Most users choose "Ask App Not to Track."

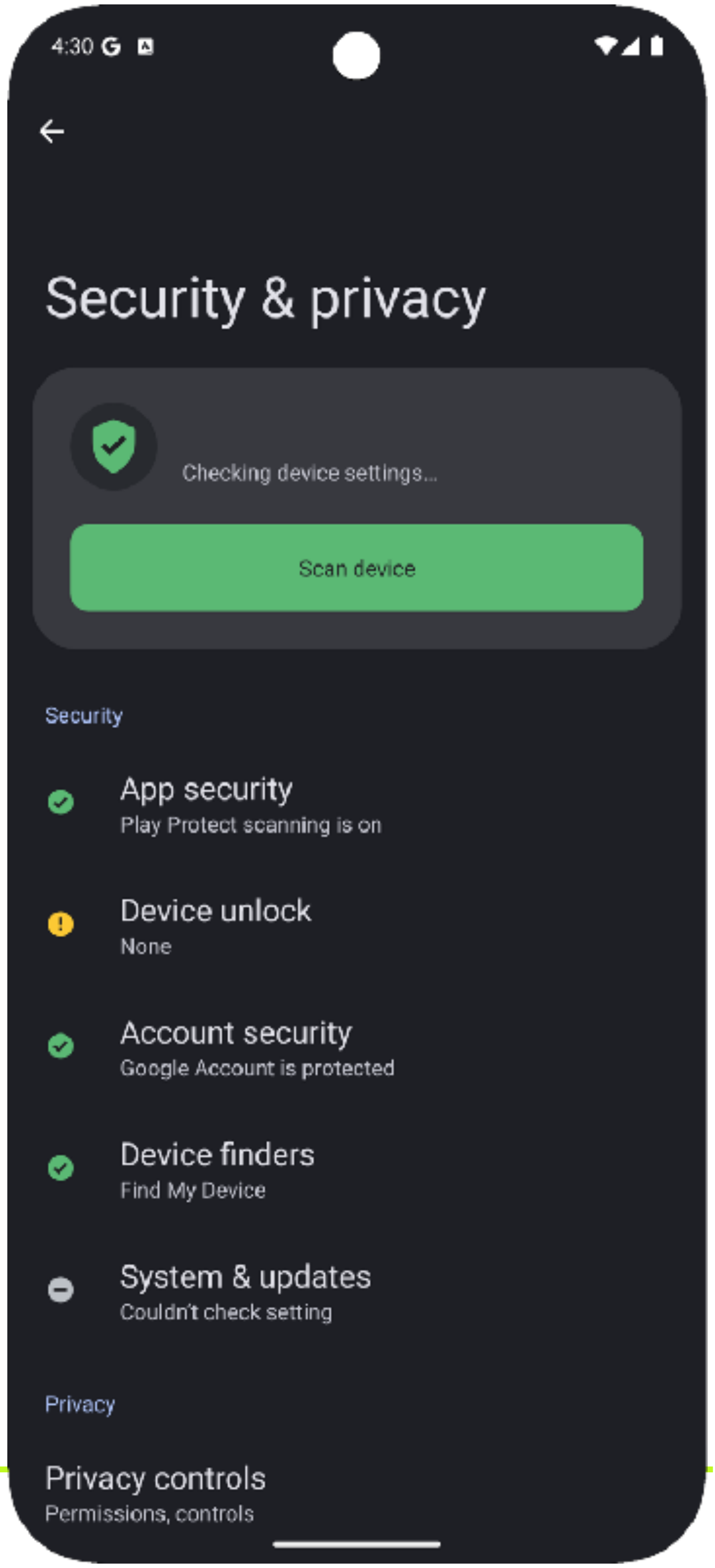


The Privacy Dashboards

Both platforms now offer a centralized place for users to review and manage permissions.

- **iOS: App Privacy Report**
  - Shows a 7-day summary of which sensors and data an app has accessed.
  - Also shows which network domains an app has contacted.
- **Android: Privacy Dashboard**
  - Shows a 24-hour timeline of when apps accessed location, camera, and microphone.
  - Provides a central place to manage all app permissions.



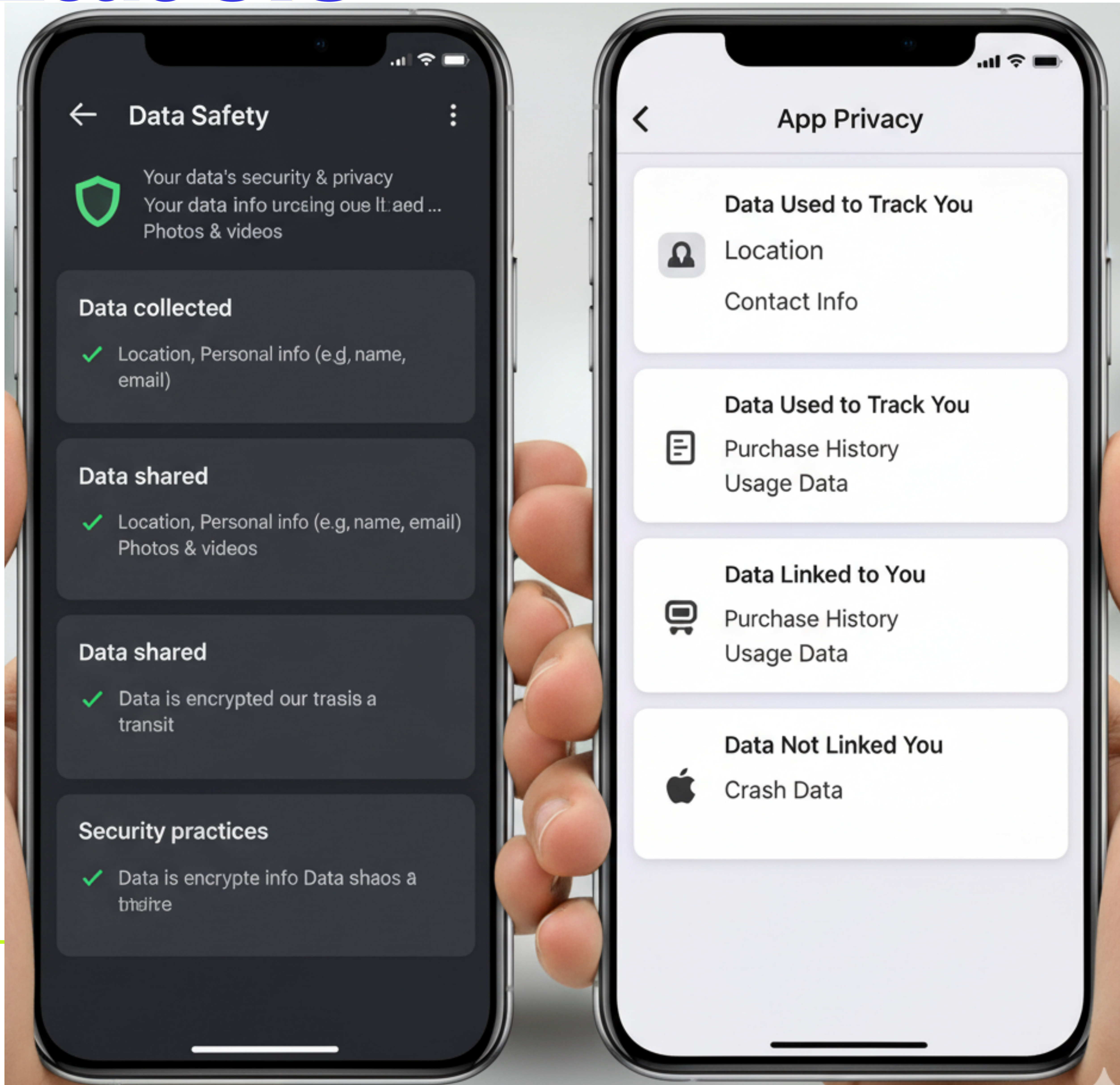


App Store Privacy Labels

Both stores now require developers to self-report their data collection practices in a simple, easy-to-read format.

- **Google Play: Data Safety Section**
- **Apple App Store: App Privacy "Nutrition Labels"**

These labels show what data an app collects, whether it's linked to you, and whether it's used for tracking.



Part 3: The Privacy Toolkit

An Introduction to Privacy-Enhancing Technologies (PETs)

What are PETs?

Privacy-Enhancing Technologies are systems and methods that minimize the amount of personal data collected, or maximize the security and confidentiality of the data that is collected.

The Goal: To enable data to be used for a specific purpose (like analytics or research) without revealing sensitive information about individuals.

Examples:

- Homomorphic Encryption
- Zero-Knowledge Proofs
- **Federated Learning**
- **k-Anonymity**
- **Differential Privacy**

PET Example 1: On-Device Intelligence

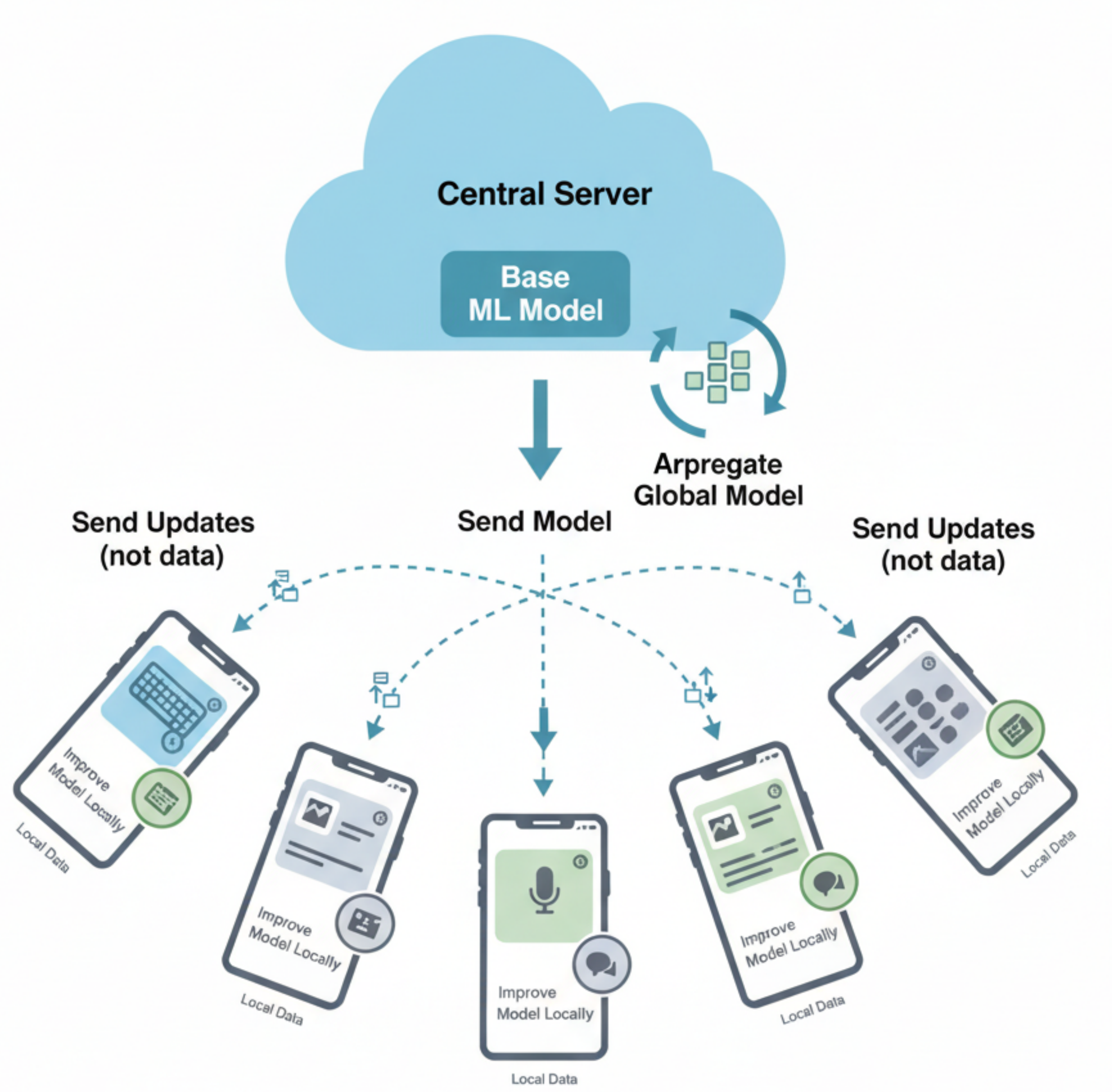
The most effective PET is to **not collect the data in the first place.**

- **The Principle:** Perform machine learning and other data processing directly on the user's device, rather than sending raw data to a server.
- **Apple's Approach:** Apple heavily promotes on-device intelligence. Features like photo recognition ("find all my pictures of dogs"), keyboard suggestions, and health monitoring are all done locally on your iPhone.
- Only the results, or anonymized data, ever leave the device.

PET Example 2: Federated Learning

A collaborative machine learning approach that doesn't require raw data to be sent to a central server.

- **How it works:**
  - A central server starts with a generic ML model.
  - The model is sent to individual devices.
  - Each device improves the model using local data (e.g., your typing patterns to improve the keyboard).
  - Each device sends only the small, anonymized updates back to the server.
  - The server aggregates thousands of these updates to improve the global model.



PET Example 3: Data Anonymization

If you must collect data, you should try to remove all Personally Identifiable Information (PII).

- **PII Examples:** Name, address, phone number, email, device ID.
- **The Process:** Stripping or hashing PII before the data is stored or analyzed.
- **The Problem:** True anonymization is extremely difficult. Seemingly non-sensitive data points can be combined to re-identify individuals. This is called a **linkage attack**.

The Linkage Attack: A Famous Example

- In the 1990s, a health insurance group in Massachusetts released "anonymized" hospital visit data for researchers.
- They stripped all obvious identifiers like name and address.
- A graduate student, Latanya Sweeney, knew the governor of Massachusetts lived in Cambridge. She bought the public Cambridge voter roll for $20.
- By cross-referencing the "anonymous" health data with the public voter data using just **ZIP code, birth date, and gender**, she was able to successfully re-identify the governor's health records.

Part 4: Hiding in the Crowd

Formal Privacy Model 1: k-Anonymity

The Goal of k-Anonymity

A dataset is k-anonymous if for any person in the dataset, there are at least k-1 other people who share the exact same set of identifying attributes.

- The identifying attributes are called **Quasi-Identifiers** (QIs). These are the fields that could be used in a linkage attack (e.g., ZIP code, birth date, gender).
- The goal is to make each individual "hide in a crowd" of size k.

Achieving k-Anonymity: An Example

Original (Vulnerable) Data:

Quasi-Identifiers: ZIP Code, Age. **Problem:** Everyone is unique. This dataset has k=1.

Name	ZIP Code	Age	Disease
Alice	12345	28	Flu
Bob	12345	29	Acute trauma
Carol	54321	41	Flu
Dave	54321	42	Flu

Techniques for k-Anonymity

To make the data k-anonymous, we must modify it to create ambiguity.

- **Generalization:** Replace specific values with more general ones.
  - *Age: 28 -> Age: 20-30*
  - ZIP: 12345 -> *ZIP: 123***
- ***Suppression:** Remove some data points entirely.*
  - Delete a record that is too unique and cannot be generalized.

ZIP Code	Age Range	Disease
123**	20-30	Flu
123**	20-30	Acute trauma
543**	40-50	Flu
543**	40-50	Flu

The k-Anonymous Result (k=2)

Anonymized Data (k=2):

Analysis:

- Now, if you search for a 20-30 year old in ZIP code 123**, you find two records (Alice and Bob). You can't tell which one has the flu and which one has acute trauma.
- Each person is now indistinguishable from at least one other person. The dataset is **2-anonymous**.

ZIP Code	Age Range	Disease
123**	20-30	Flu
123**	20-30	Acute trauma
543**	40-50	Flu
543**	40-50	Flu

Limitations of k-Anonymity

k-Anonymity is a good start, but it has weaknesses.

- **Homogeneity Attack:** If all the values for the sensitive attribute within a group are the same, privacy is breached.

  Example: If both Alice and Bob had the flu, you would know that any 20-30 year old in ZIP 123 has the flu.*

- **Background Knowledge Attack:** An attacker might have outside information that allows them to defeat the anonymity.

  Example: The attacker knows that Bob is an avid cyclist and is very unlikely to have the flu. They can deduce he is the one with acute trauma.

Beyond k-Anonymity: l-Diversity

To solve the homogeneity problem, **l-diversity** was proposed.

- **Principle:** In addition to being k-anonymous, every group of records must also have at least *l* distinct values for the sensitive attribute.
- **Example (l=2):** In our previous example, the group *{Flu, Trauma}* is 2-diverse. The group *{Flu, HIV}* is also 2-diverse. If a group was *{Flu, Flu}*, it would fail the l-diversity test.
- This prevents an attacker from learning the sensitive value even if they can identify the group.

The Curse of Dimensionality

A major challenge for k-anonymity and its variants is the "Curse of Dimensionality."

- **The Problem:** As you add more quasi-identifier attributes (e.g., ZIP, age, gender, marital status, number of children...), it becomes exponentially harder to form groups.
- The data becomes "sparse," and almost everyone becomes unique again.
- To achieve k-anonymity, you have to generalize the data so much that it becomes useless for analysis.

Part 5: Statistical Secrecy

Formal Privacy Model 2: Differential Privacy (DP)

The Core Idea of Differential Privacy

A query or analysis is differentially private if its output does not significantly change when a single individual's data is added to or removed from the dataset.

The Goal: To learn useful patterns about the group while learning nothing about any specific individual.

The Guarantee: A person's privacy is protected because their participation in the dataset has a negligible effect on the final result. They have **plausible deniability**.

An Analogy: The Secret Cookie Jar

Imagine a class of 30 students. The teacher wants to know how many students have ever took a cookie from the cookie jar, but no one wants to admit it.

The DP Protocol:

- The teacher tells each student to flip a coin in secret.
- **If it's tails:** You must answer truthfully ("Yes" or "No").
- **If it's heads:** Flip the coin again. If it's heads, say "Yes." If it's tails, say "No."

The Cookie Jar: Plausible Deniability

Now, if a student answers "Yes," what does it mean?

- It could mean they took a cookie.
- It could also mean they got heads on the first flip and heads on the second flip.

No one can be accused, because every answer has **plausible deniability**. The randomness of the coin flips protects each individual.

The Cookie Jar: The Magic

But how can the teacher get a useful result?

- The teacher knows the probability of the coin flips. They know that roughly 25% of the class will say "Yes" just due to random chance (Heads, then Heads).
- Let's say 17 students (57%) answered "Yes."
- The teacher can subtract the expected noise (25% of 30 = 7.5 students) from the total "Yes" count.
- **Result:** The teacher can estimate that roughly 9-10 real cookie-stealers exist, **without knowing the identity of any single one.**

The Mechanism: Adding Controlled Noise

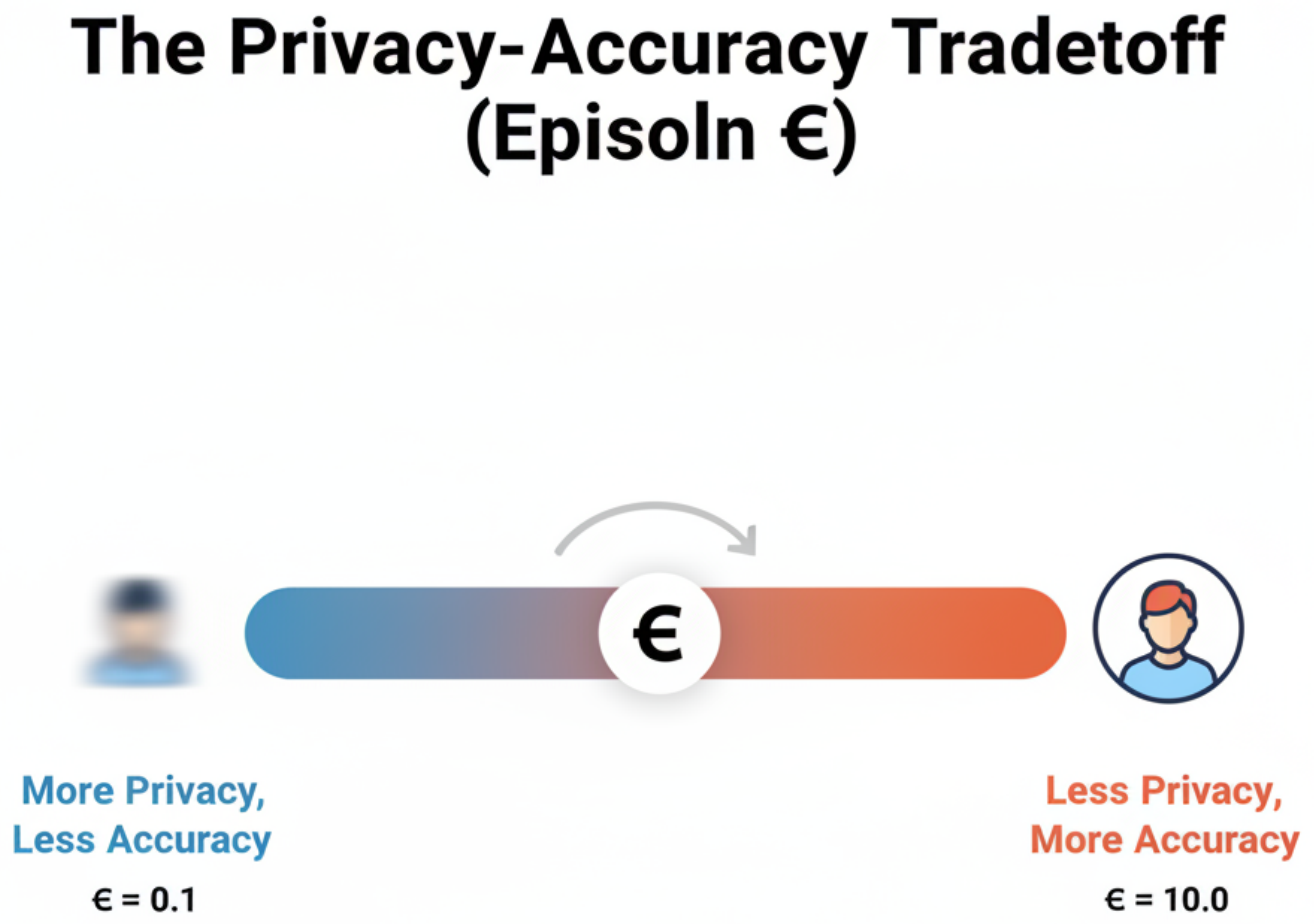
Differential Privacy is achieved by adding carefully calibrated statistical noise to the results of a query.

- The amount of noise added is controlled by a parameter called **epsilon (ε)**, also known as the **privacy budget**.
  - **Low ε (e.g., 0.1):** Lots of noise, high privacy, low accuracy.
  - **High ε (e.g., 8):** Little noise, low privacy, high accuracy.

The Privacy Budget (ε)

Epsilon is a measure of how much the output can change if one person's data is changed.

- Each query "spends" some of the privacy budget.
- Once the total budget is used up for a dataset, no more queries can be run to prevent revealing too much information.
- Choosing the right epsilon is one of the hardest parts of implementing DP.



The Laplace Mechanism

One of the most common ways to achieve DP for numerical queries (like counts or sums) is the **Laplace Mechanism**.

- **The Process:**
- Calculate the true result of the query (e.g., "How many users visited this location?").
- Determine the **sensitivity** of the query. This is the maximum amount the result could change if one person's data was removed.
- Generate a random number from a **Laplace distribution**, scaled by the sensitivity and the privacy budget (ε).
- Add this random number to the true result.
- Return the noisy result.

Laplace Mechanism in Code (Conceptual)

This is what the Laplace Mechanism looks like in code.

```python
import numpy as np

def laplace_mechanism(true_result, sensitivity, epsilon):
    """
    Adds Laplace noise to a numerical result to achieve differential privacy.
    """
    # The scale of the noise is determined by sensitivity / epsilon
    scale = sensitivity / epsilon

    # Generate a random number from the Laplace distribution
    noise = np.random.laplace(loc=0, scale=scale)

    # Return the noisy result
    return true_result + noise
```

```python
# --- Example ---
# Query: How many users are in New York?
true_user_count = 5250
# Sensitivity is 1 (one user can change the count by at most 1)
query_sensitivity = 1
# We choose a privacy budget of 0.1
privacy_budget_epsilon = 0.1

# Run the query 5 times
for i in range(5):
    noisy_count = laplace_mechanism(true_user_count, query_sensitivity, privacy_budget_epsilon)
    print(f"Noisy Result {i+1}: {noisy_count:.2f}")

# Expected Output:
# Noisy Result 1: 5241.75
# Noisy Result 2: 5262.31
# Noisy Result 3: 5255.19
# Noisy Result 4: 5239.88
# Noisy Result 5: 5258.90
```

Laplace Mechanism in Action

Query: "How many users in our database live in ZIP code 12345?"

- **True Answer:** 100
- **Sensitivity:** 1 (If we add or remove one person, the count changes by at most 1).
- **Privacy Budget (ε):** Let's choose $\varepsilon = 0.1$ (high privacy).
- **Laplace Noise:** The mechanism generates a random number from the Laplace distribution scaled by $sensitivity / \varepsilon = 1 / 0.1 = 10$. Let's say the random number is *-8.3*.
- **Final Answer:** $100 + (-8.3) = 91.7$

The system reports **92**.

DP in the Real World: Apple & Google

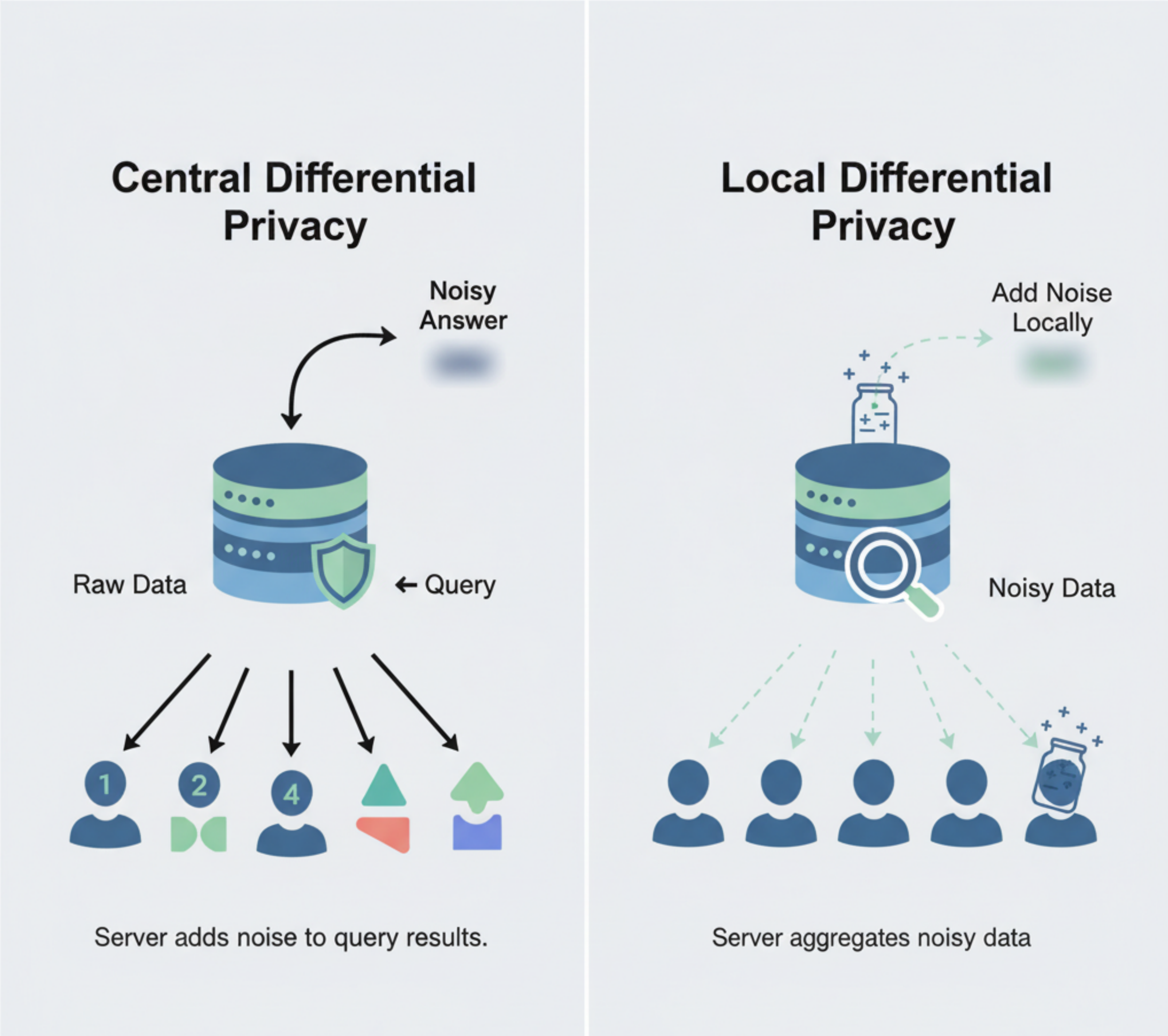
Differential Privacy is not just a theoretical concept. It's used to protect the data of billions of people.

- **Apple:** Uses DP to collect data about keyboard suggestions, emoji usage, and health data without reading your personal content.
- **Google:** Uses DP to collect telemetry from the Chrome browser and to provide real-time traffic data in Google Maps.
- **The US Census Bureau:** Used DP to protect the privacy of respondents in the 2020 Census.

Local vs. Central Differential Privacy

There are two main models for applying DP:

- **Central DP:** A trusted curator collects the raw, sensitive data and then adds noise to the answers to queries. (Our database example).
- **Local DP:** Noise is added to each individual's data on their own device before it is ever sent to a server. The server never sees the true, raw data. (Our cookie jar analogy).



Local DP in Code (Conceptual)

This is how an app could implement the "cookie jar" protocol (a form of Local DP) before sending data to a server.

```
func getNoisyAnswer(trueAnswer: Bool) -> Bool {
    // This is the Local DP protocol.
    // p is the probability of answering truthfully.
    // q is the probability of answering "Yes" randomly.

    // Flip the first coin
    if Bool.random() { // 50% chance
        // Answer truthfully
        return trueAnswer
    } else { // 50% chance
        // Answer randomly
        return Bool.random()
    }
}
```

```
// Usage:
let didUserEnableFeature = true // The user's real, private data
let noisyDataToSend = getNoisyAnswer(trueAnswer: didUserEnableFeature)

// Send `noisyDataToSend` to the server.
// The server never sees the value of `didUserEnableFeature`.
```

Part 6: The Road Ahead

Summary and What's Next

Key Takeaways (1/3)

- **The Mobile Data Ecosystem is Vast:** Your data is a valuable commodity, and a complex web of trackers and brokers exists to collect and trade it. Fingerprinting is the new frontier in the tracking arms race.
- **Permissions are the First Line of Defense:** Modern OS permission systems (runtime, one-time, auto-reset) and transparency features (dashboards, labels) give users more granular control than ever before.

Key Takeaways (2/3)

- **Simple Anonymization is Not Enough:** Linkage attacks can easily re-identify individuals in supposedly anonymous datasets. This is why we need PETs.
- **Formal Models Provide Proof:** k-Anonymity ensures you can hide in a crowd, but has limitations. It's a useful first step for preventing simple linkage attacks.

Key Takeaways (3/3)

- **Differential Privacy is the Gold Standard:** By adding calibrated noise, DP allows us to learn about populations while providing a mathematical guarantee of privacy for individuals.
- **Privacy has a Cost:** Achieving privacy, whether through generalization in k-Anonymity or noise in DP, requires a trade-off with data accuracy. The privacy budget (Îµ) is the tool we use to manage this trade-off.

The Big Picture: A Layered Defense

- **Outer Layer:** App Store Labels & Data Safety (Transparency)
- **Next Layer:** OS Permission System (User Control)
- **Next Layer:** On-Device Processing (Data Minimization)
- **Next Layer:** Anonymization & Formal Models (DP, k-Anonymity)
- **Core:** Secure Hardware (Protecting the raw data)

What's Next?

Developing a Corporate Mobile Security Strategy

- Bring Your Own Device (BYOD) vs. Corporate-Owned Policies.
- Mobile Device Management (MDM), Mobile Application Management (MAM), and Unified Endpoint Management (UEM).
- Implementing Mobile Security Technical Controls.
- Incident Response for Mobile Devices.

Q&A

Questions?