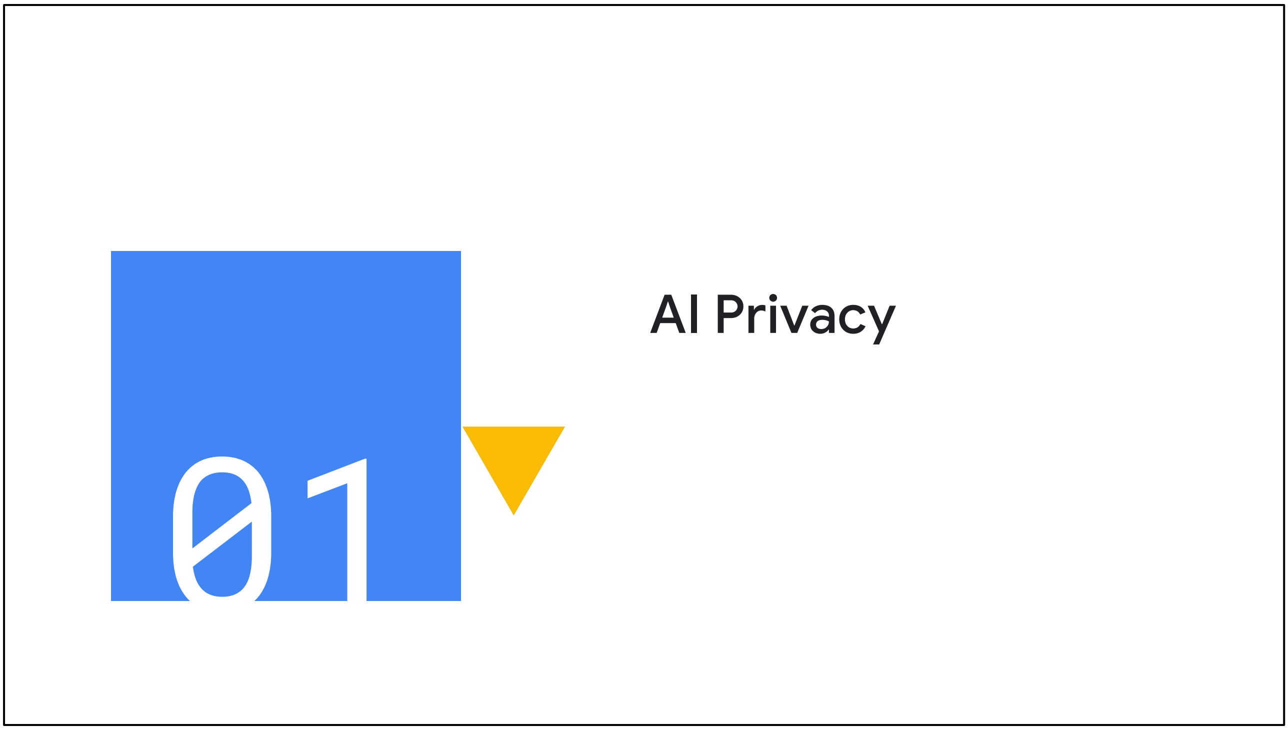
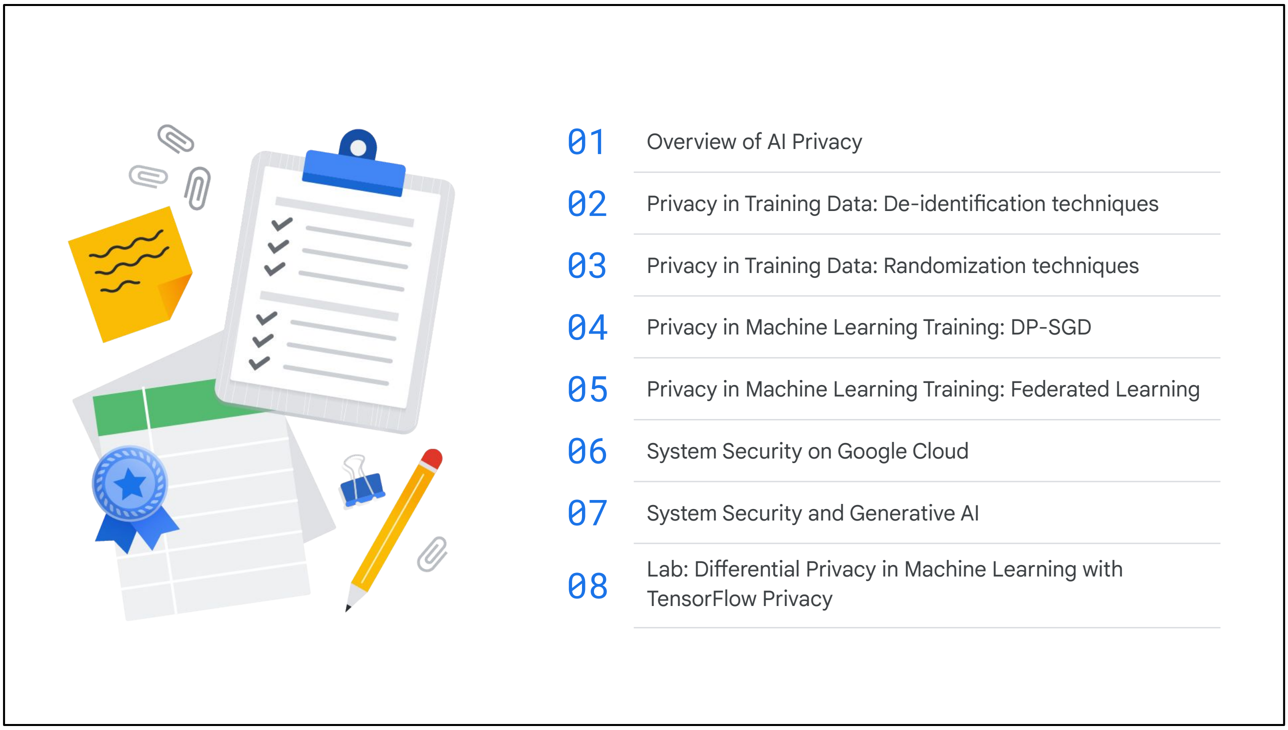


01

AI Privacy

Welcome to AI Privacy.



01 Overview of AI Privacy

02 Privacy in Training Data: De-identification techniques

03 Privacy in Training Data: Randomization techniques

04 Privacy in Machine Learning Training: DP-SGD

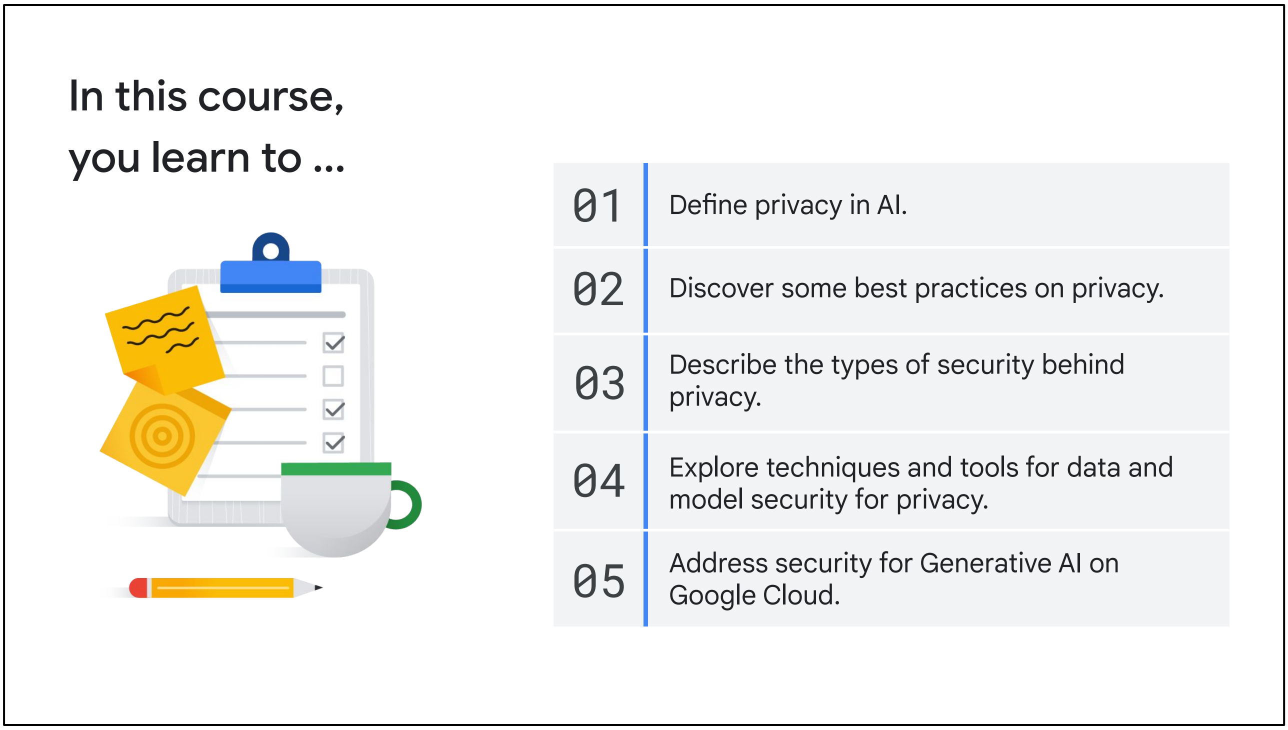
05 Privacy in Machine Learning Training: Federated Learning

06 System Security on Google Cloud

07 System Security and Generative AI

08 Lab: Differential Privacy in Machine Learning with TensorFlow Privacy

This module consists of 8 lessons.



In this course, you learn to ...

01 Define privacy in AI.

02 Discover some best practices on privacy.

03 Describe the types of security behind privacy.

04 Explore techniques and tools for data and model security for privacy.

05 Address security for Generative AI on Google Cloud.

Today you will learn to…

- Define privacy in AI.
- Discover some best practices on privacy.
- Describe the types of security behind privacy.
- Explore techniques and tools for data and model security for privacy.
- Explain how to apply security best practices for generative AI models in Google Cloud.

01 Overview of AI Privacy

02 Privacy in Training Data: De-identification techniques

03 Privacy in Training Data: Randomization techniques

04 Privacy in Machine Learning Training: DP-SGD

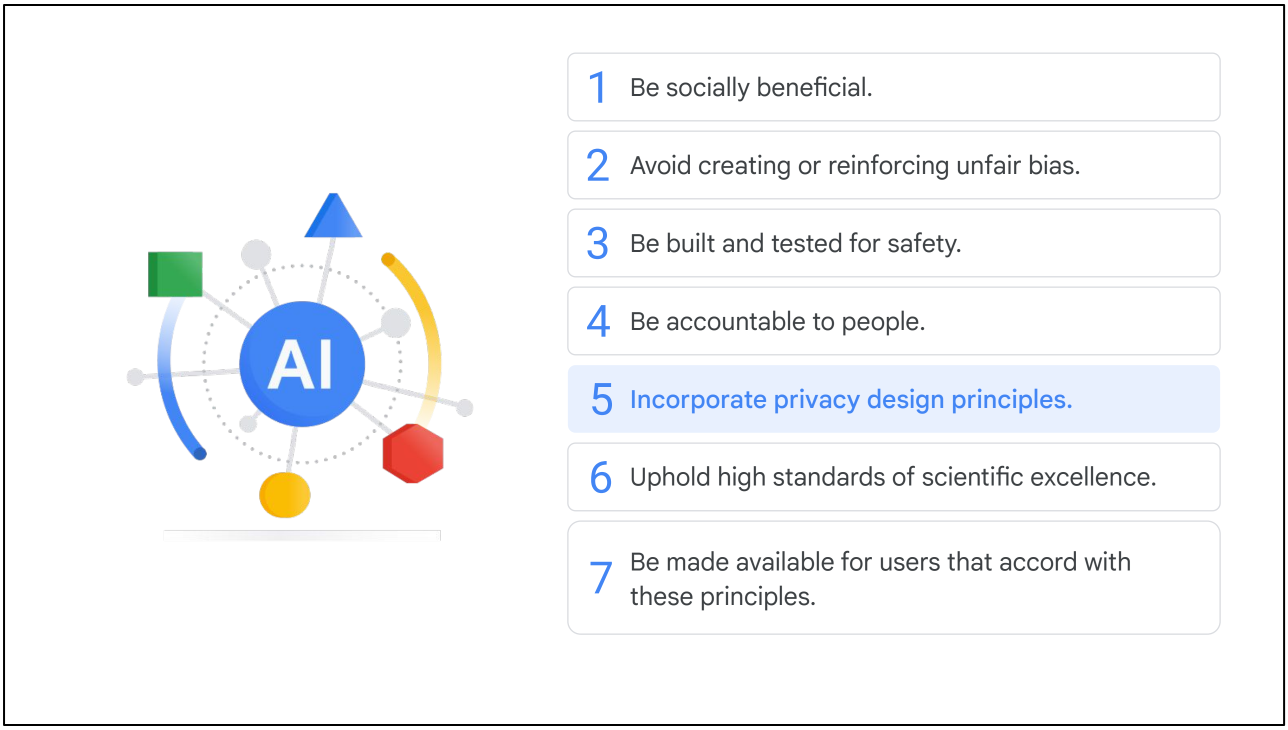
05 Privacy in Machine Learning Training: Federated Learning

06 System Security on Google Cloud

07 System security and Generative AI

08 Lab: Differential Privacy in Machine Learning with TensorFlow Privacy

Let’s start with an overview of AI Privacy.



1 Be socially beneficial.

2 Avoid creating or reinforcing unfair bias.

3 Be built and tested for safety.

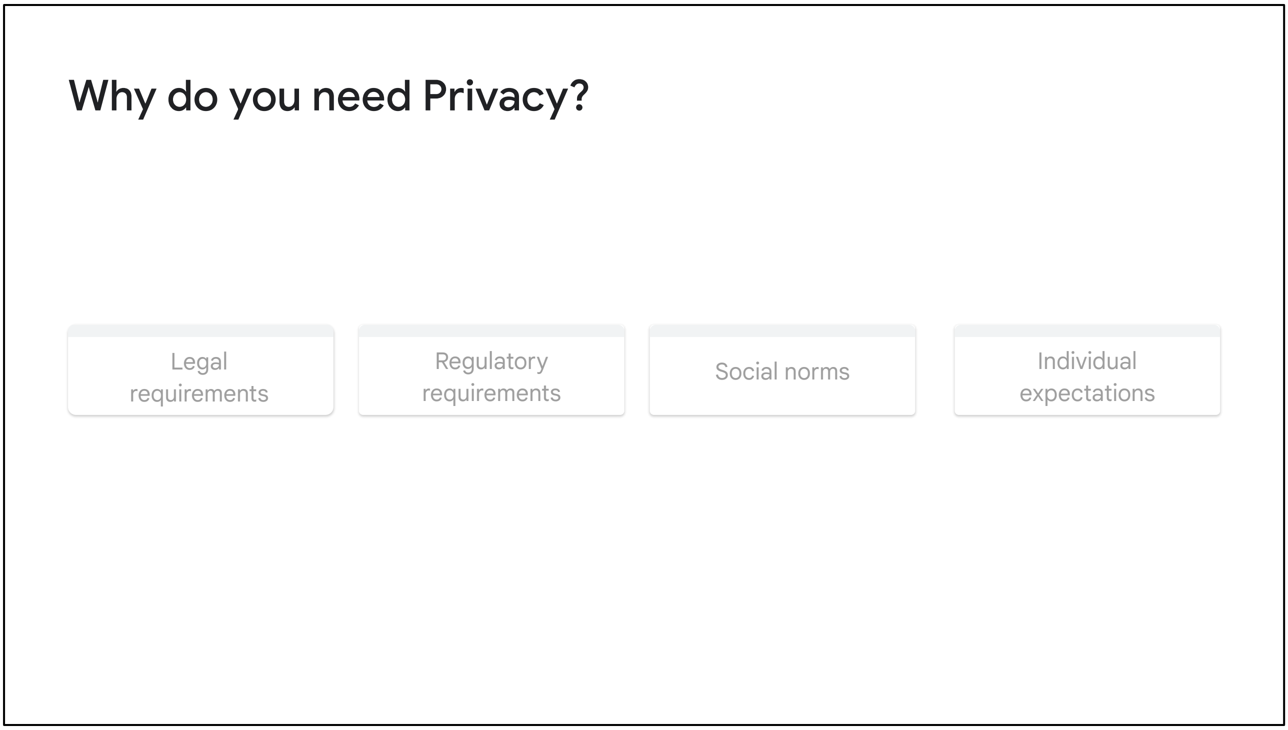
4 Be accountable to people.

5 Incorporate privacy design principles.

6 Uphold high standards of scientific excellence.

7 Be made available for users that accord with these principles.

Privacy relates to Google’s fifth AI principle: “Incorporate privacy design principles”; although it’s worth noting that the principles often intersect.



Why do you need Privacy?

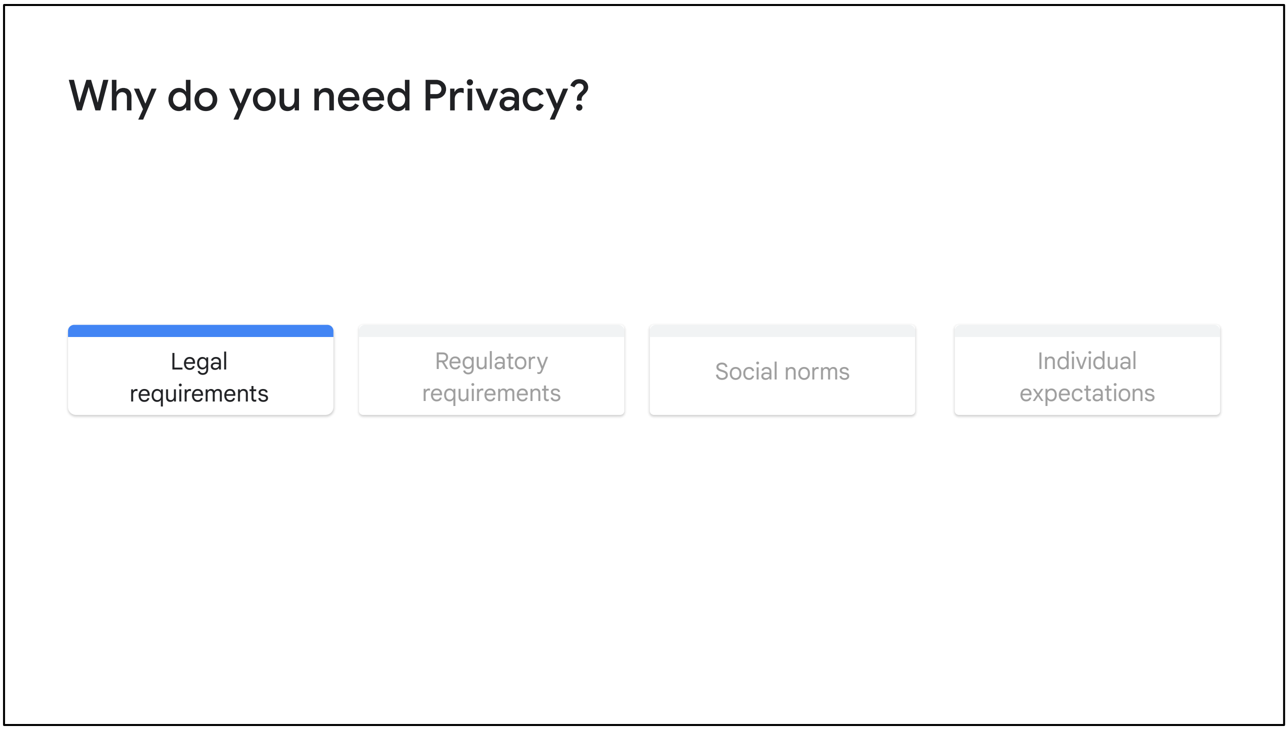
Legal requirements

Regulatory requirements

Social norms

Individual expectations

Why do you need to take privacy into consideration? There are several reasons.



Why do you need Privacy?

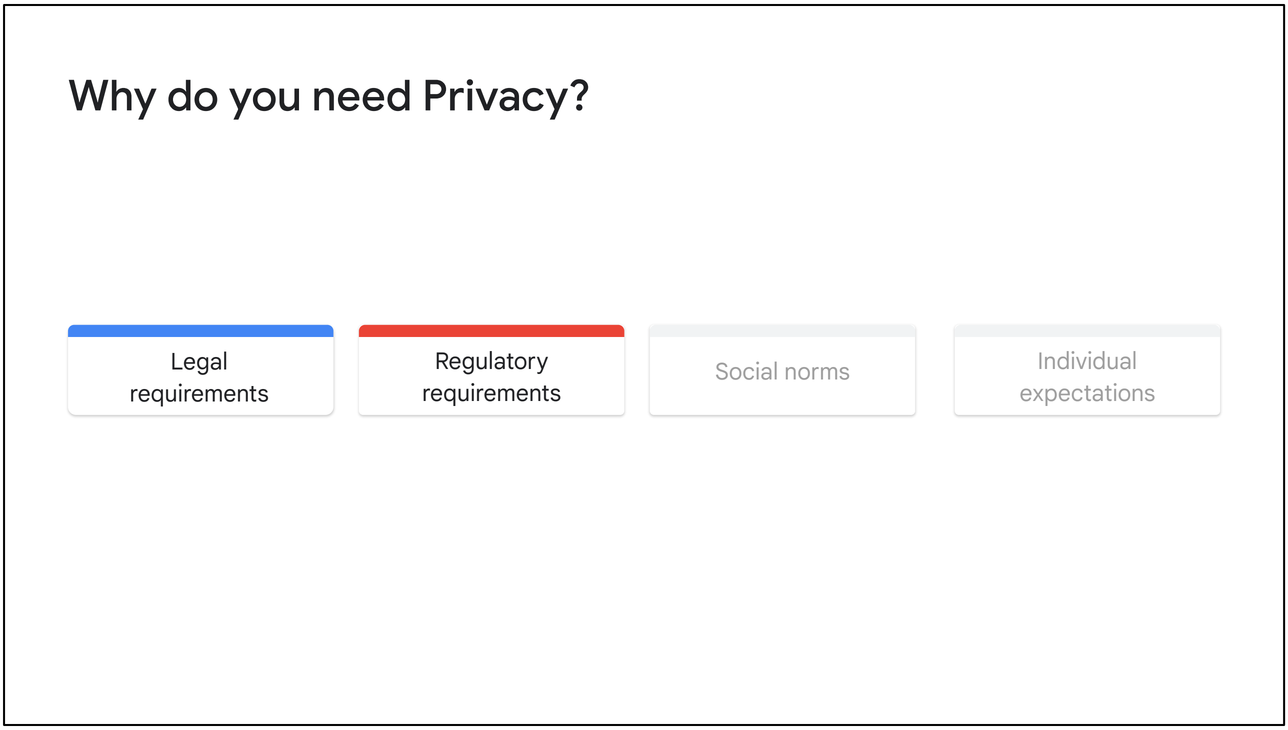
Legal requirements

Regulatory requirements

Social norms

Individual expectations

This includes not only respecting the legal



Why do you need Privacy?

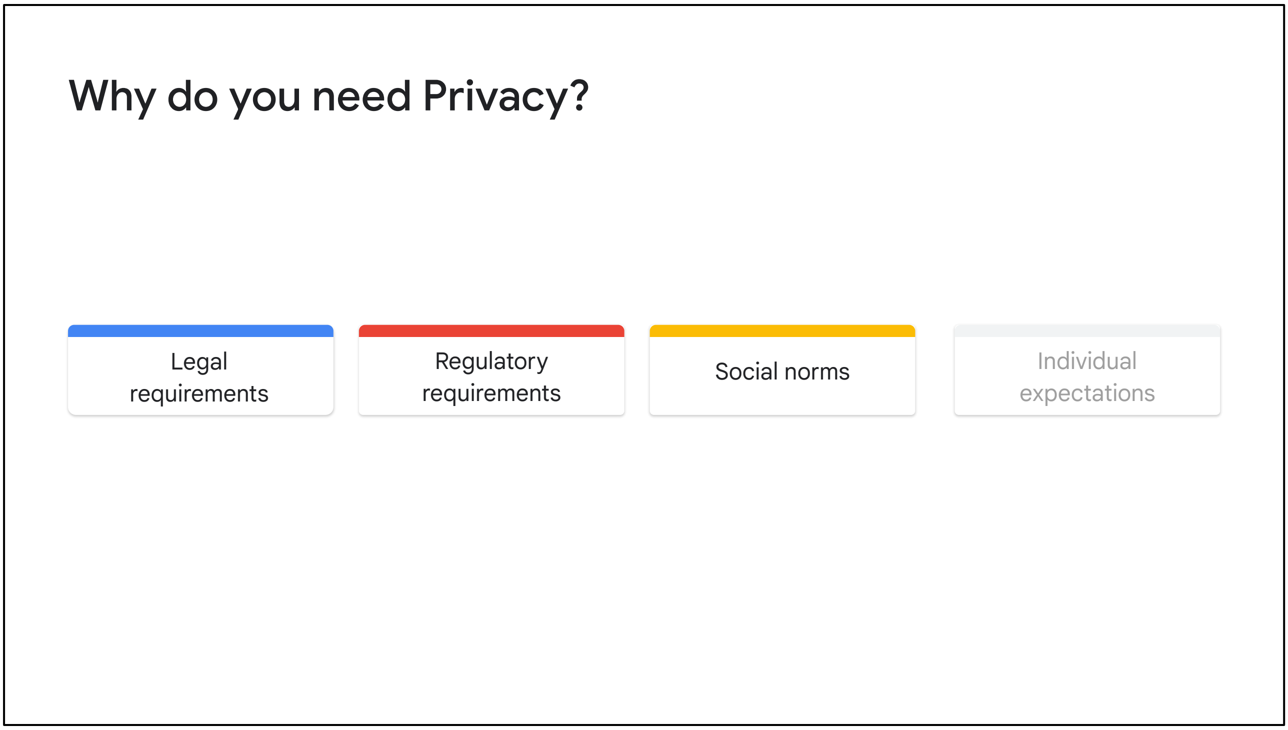
Legal requirements

Regulatory requirements

Social norms

Individual expectations

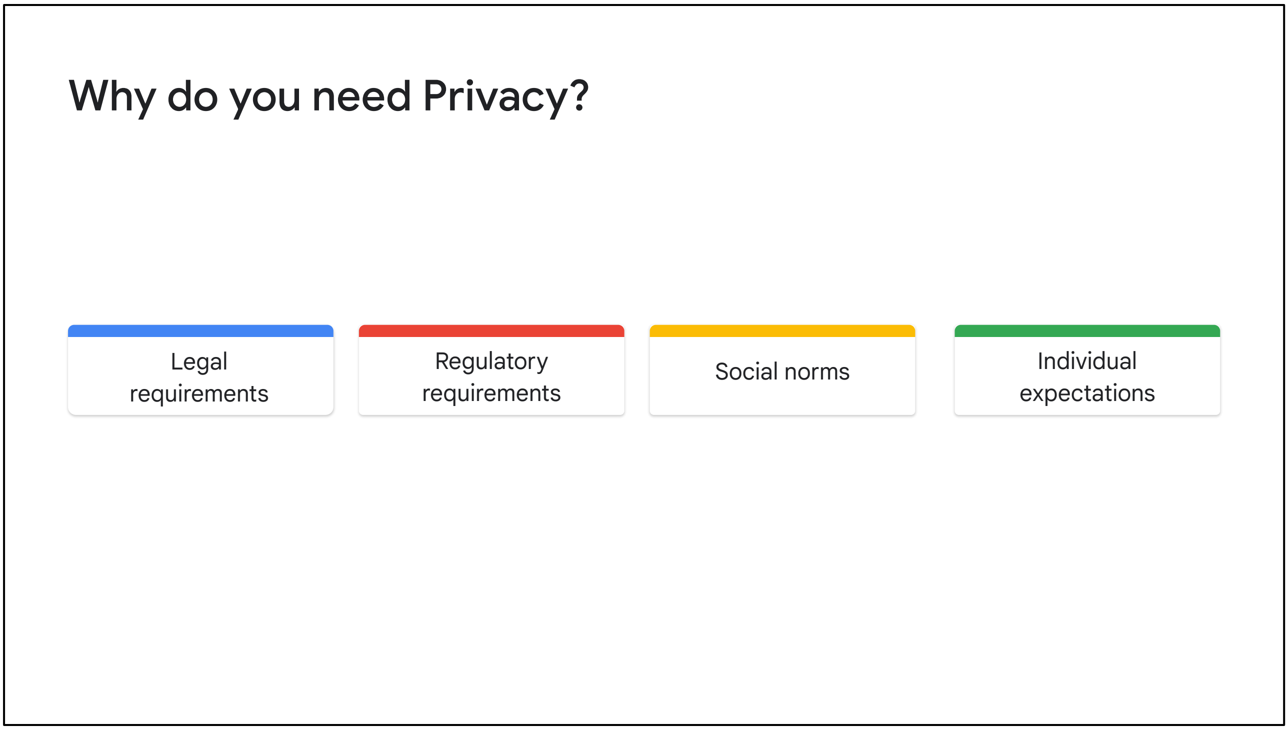
and regulatory requirements,



Why do you need Privacy?

- Legal requirements
- Regulatory requirements
- Social norms
- Individual expectations

but also considering social norms



Why do you need Privacy?

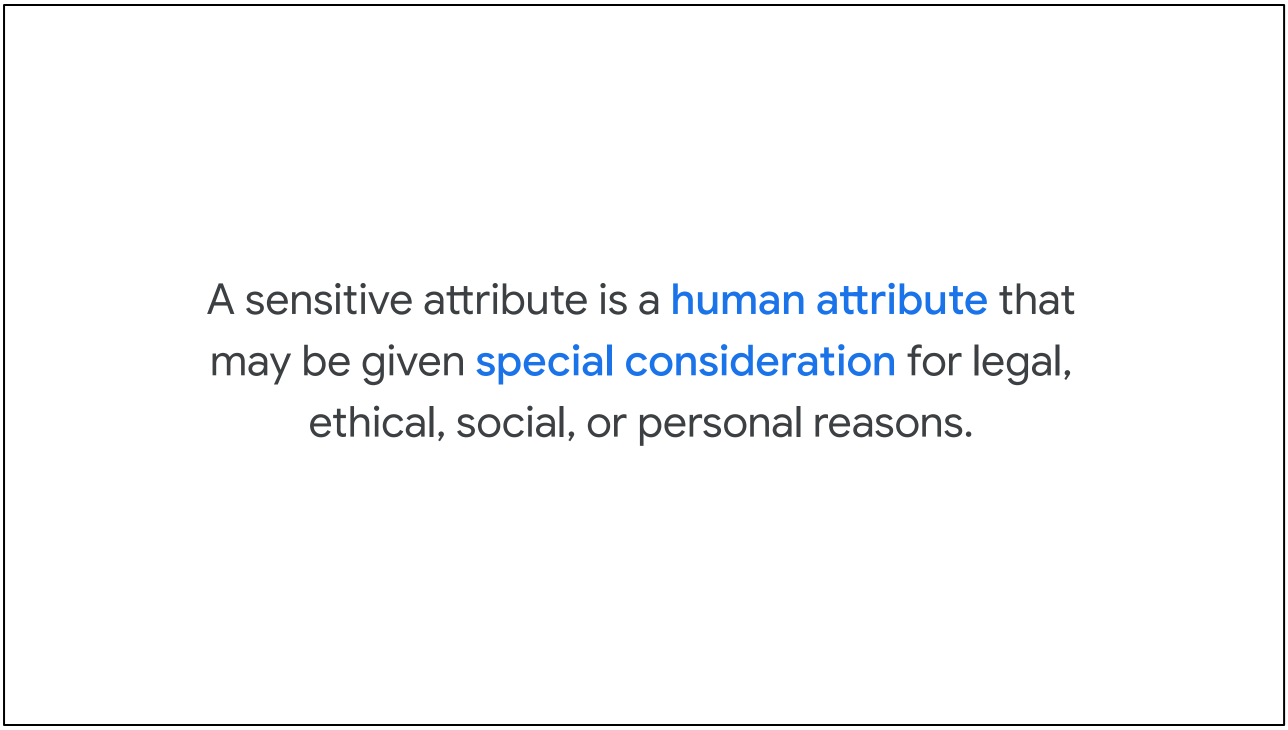
Legal requirements

Regulatory requirements

Social norms

Individual expectations

and typical individual expectations.

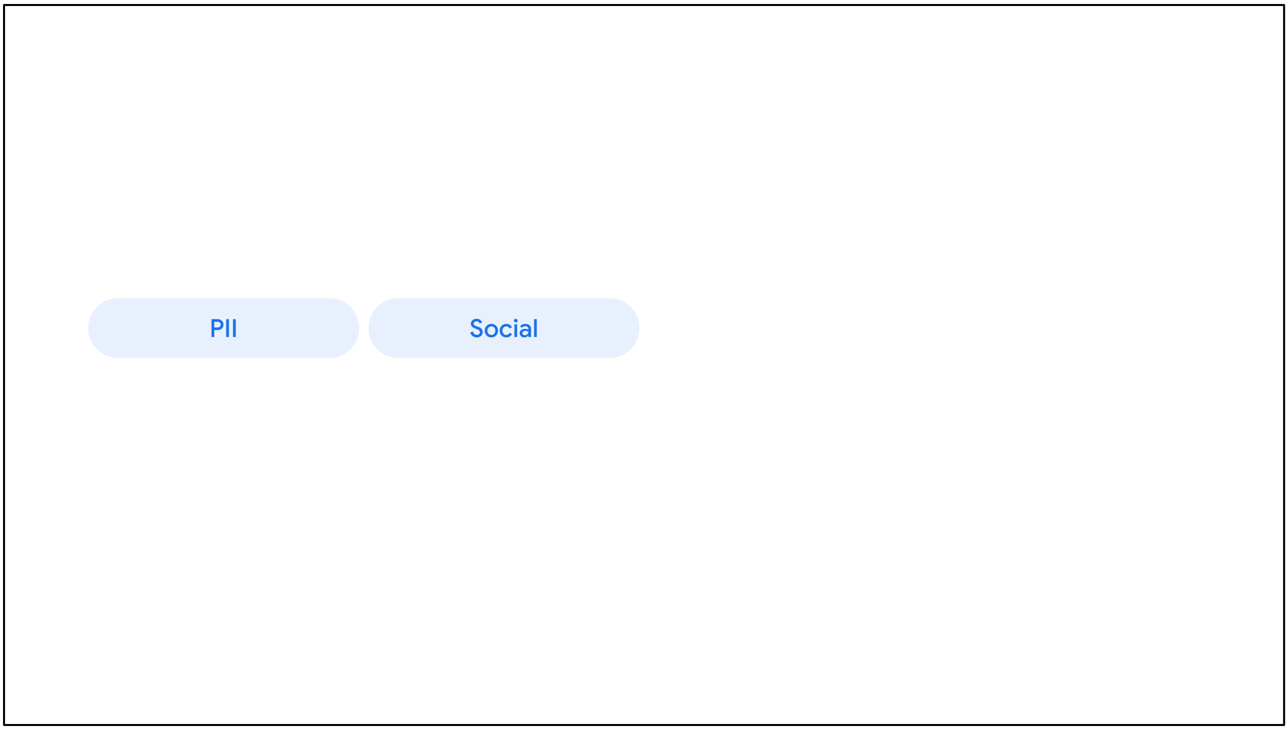


A sensitive attribute is a **human attribute** that may be given **special consideration** for legal, ethical, social, or personal reasons.

A sensitive attribute is any human attribute that may be given special consideration for legal, ethical, social, or personal reasons. This includes, but is not limited to:

PII

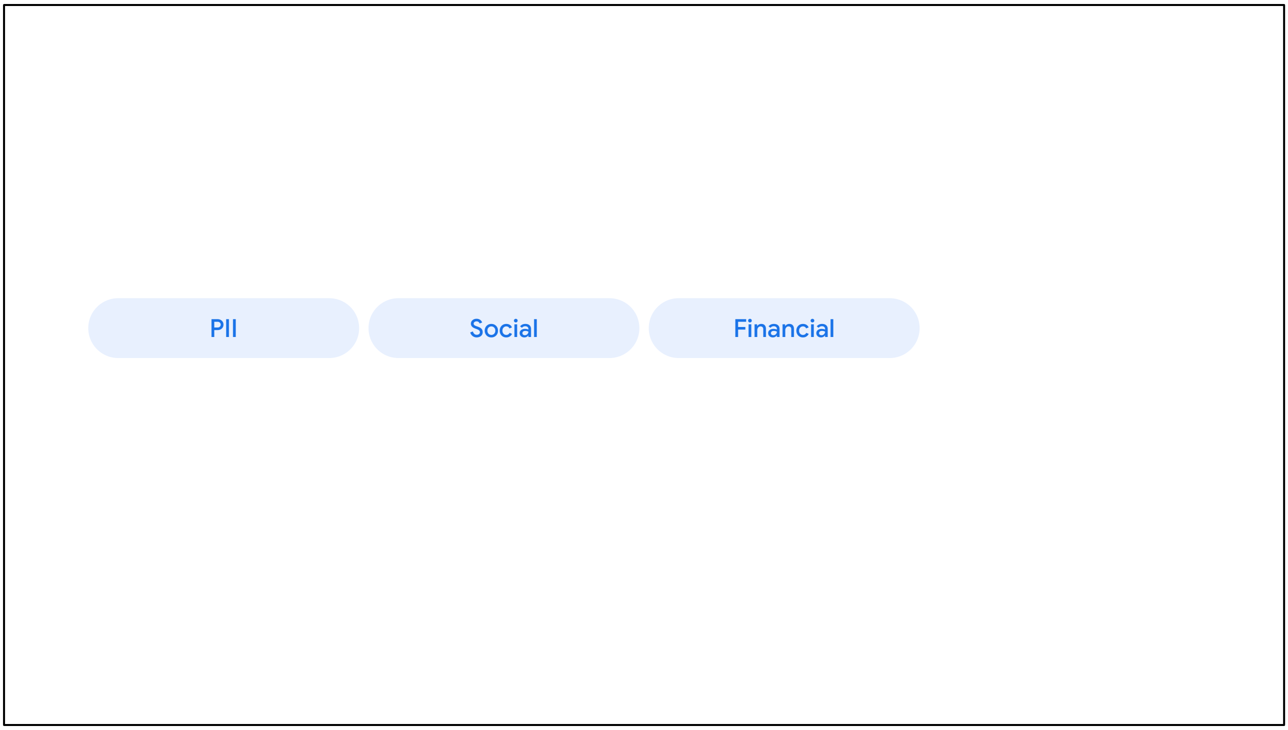
- Personal identifiable information, such as full names, date of birth, address, and phone number.



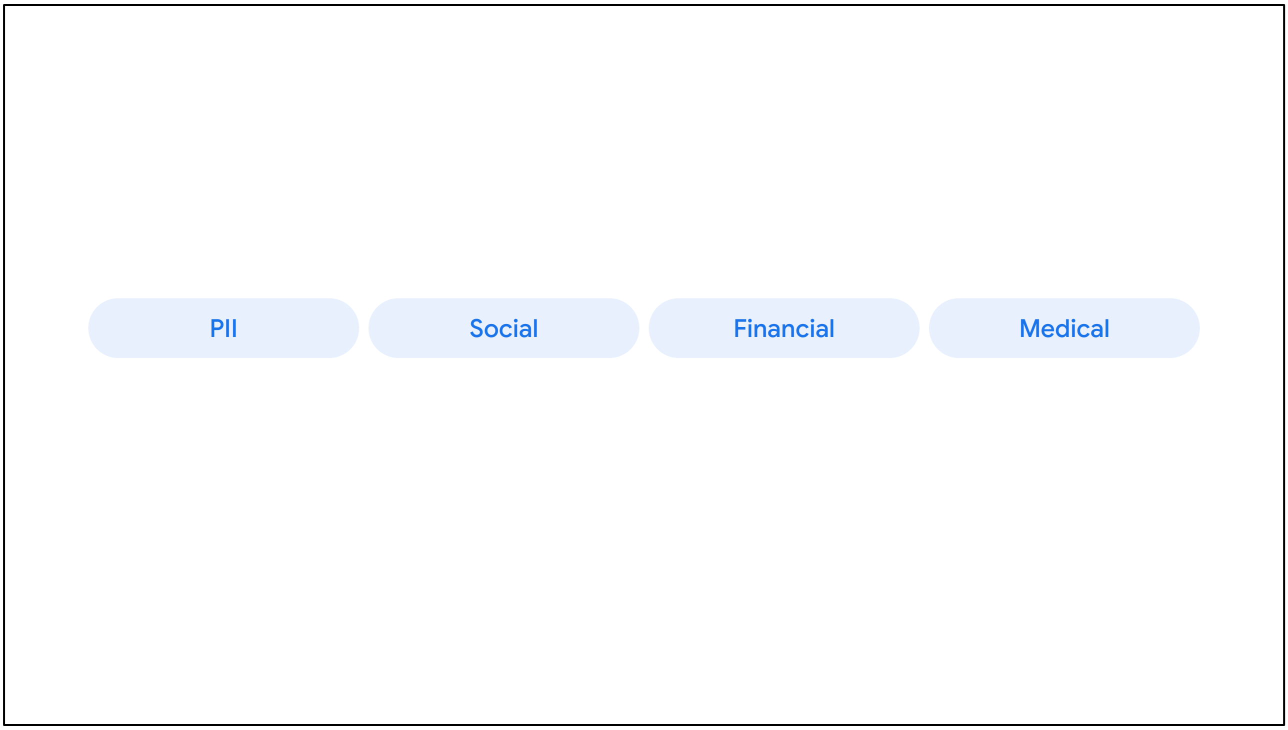
PII

Social

- Social data such as ethnicity, religion, sexual orientation, and political affiliation.

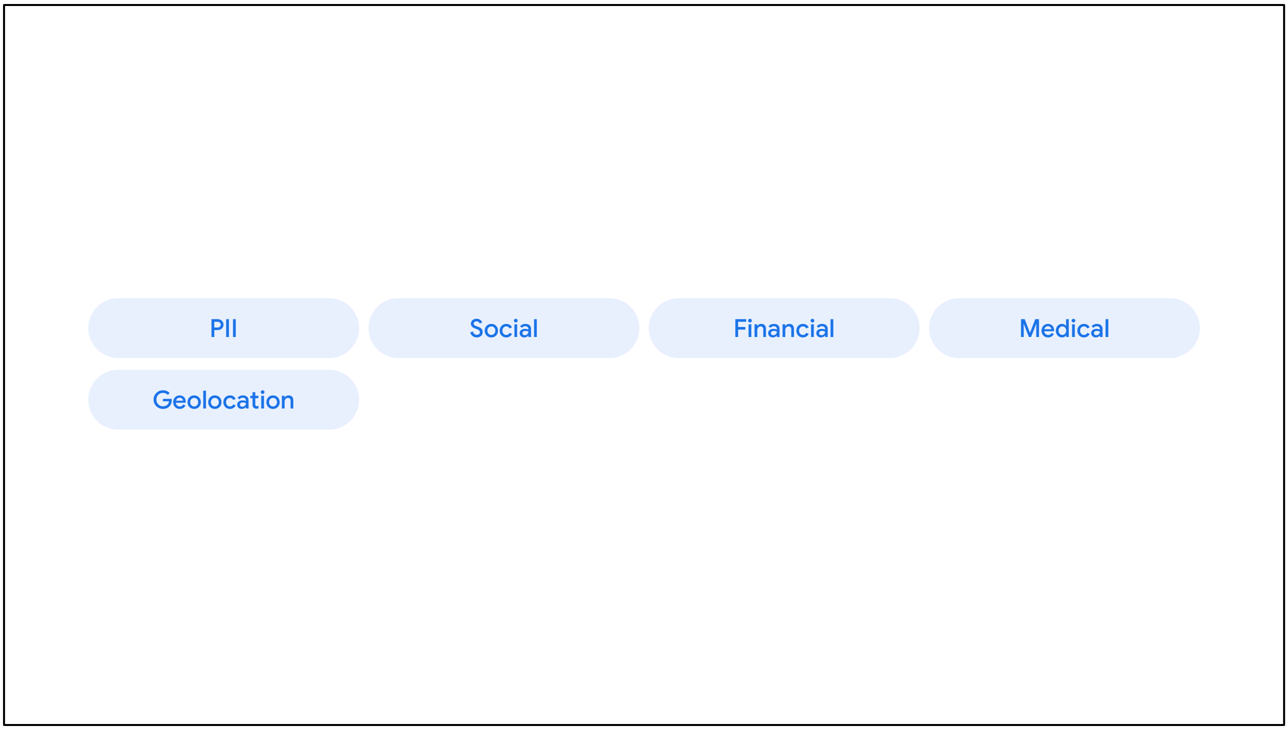


- Financial data such as credit card numbers, income, and tax records.



PII Social Financial Medical

- Health and medical data, such as diagnoses, prescriptions, and genetic data.



PII

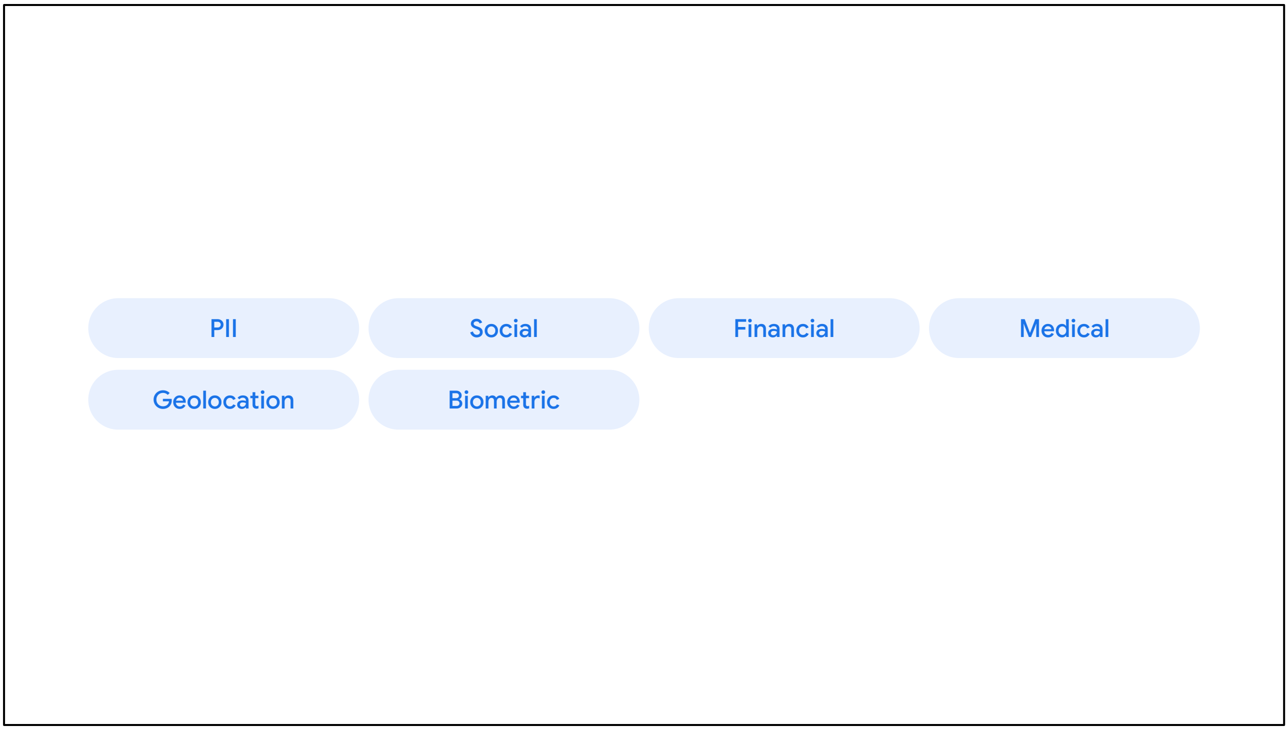
Social

Financial

Medical

Geolocation

- Geolocation data, such as tracking data from mobile devices.



PII

Social

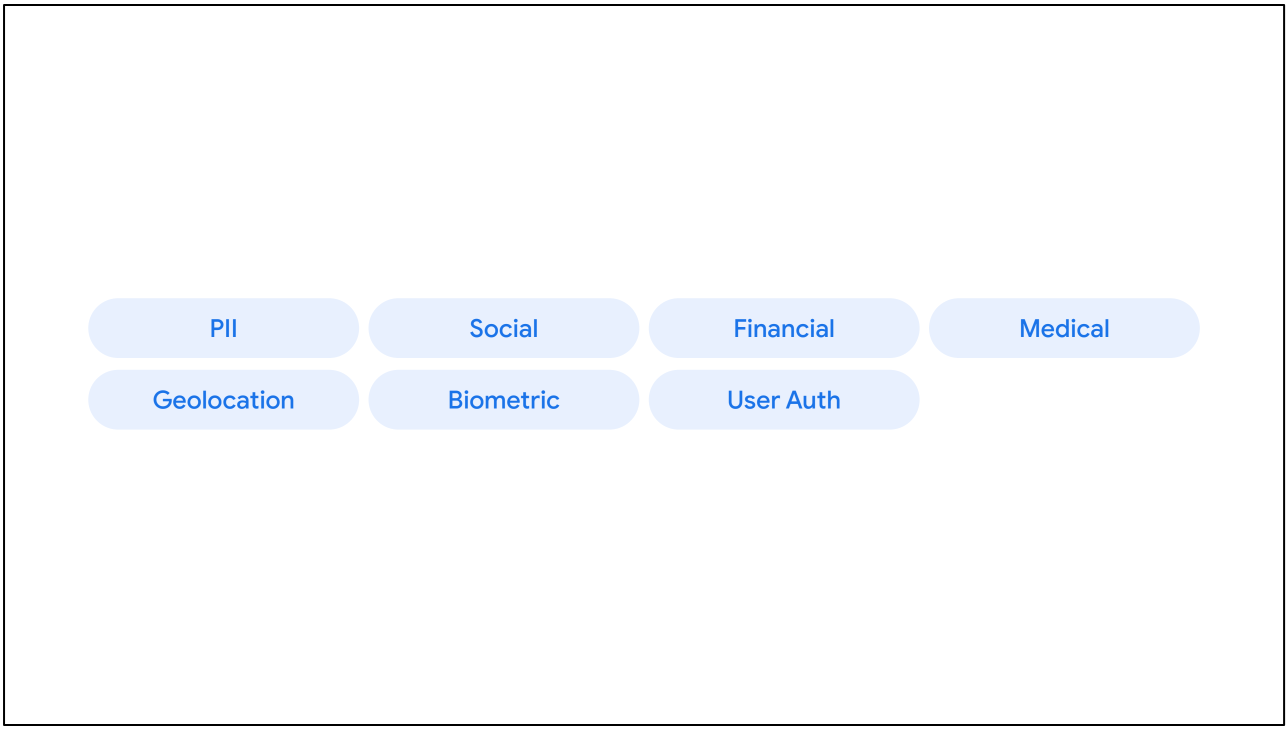
Financial

Medical

Geolocation

Biometric

- Biometric data, such as facial recognition, voice, and fingerprints.



PII

Social

Financial

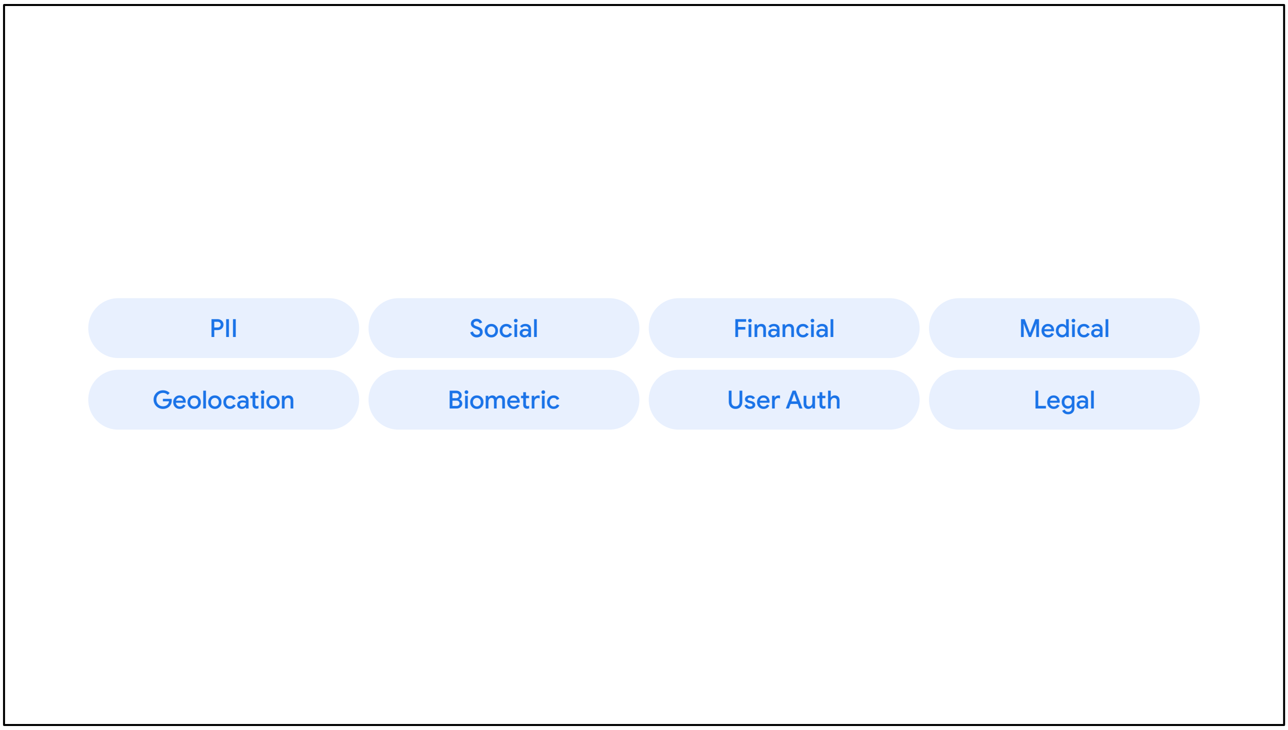
Medical

Geolocation

Biometric

User Auth

- User authentication data, such as usernames, passwords, security questions and answers.



PII

Social

Financial

Medical

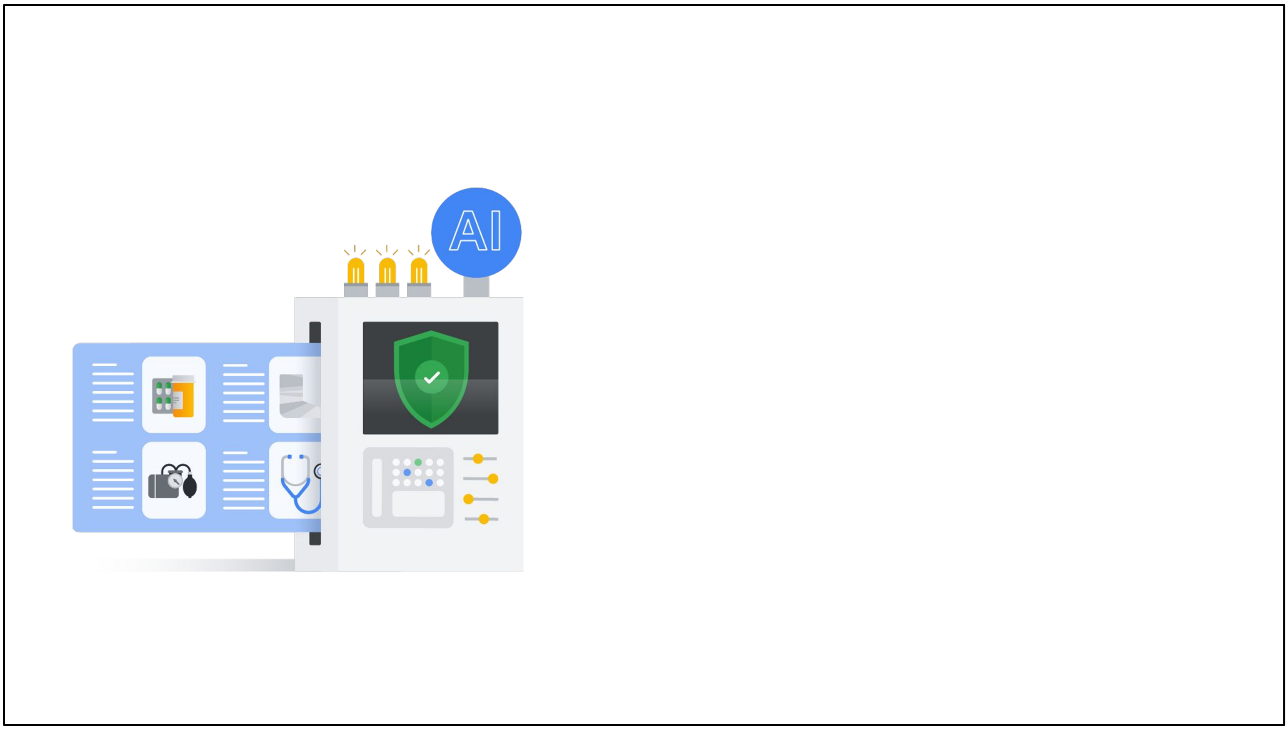
Geolocation

Biometric

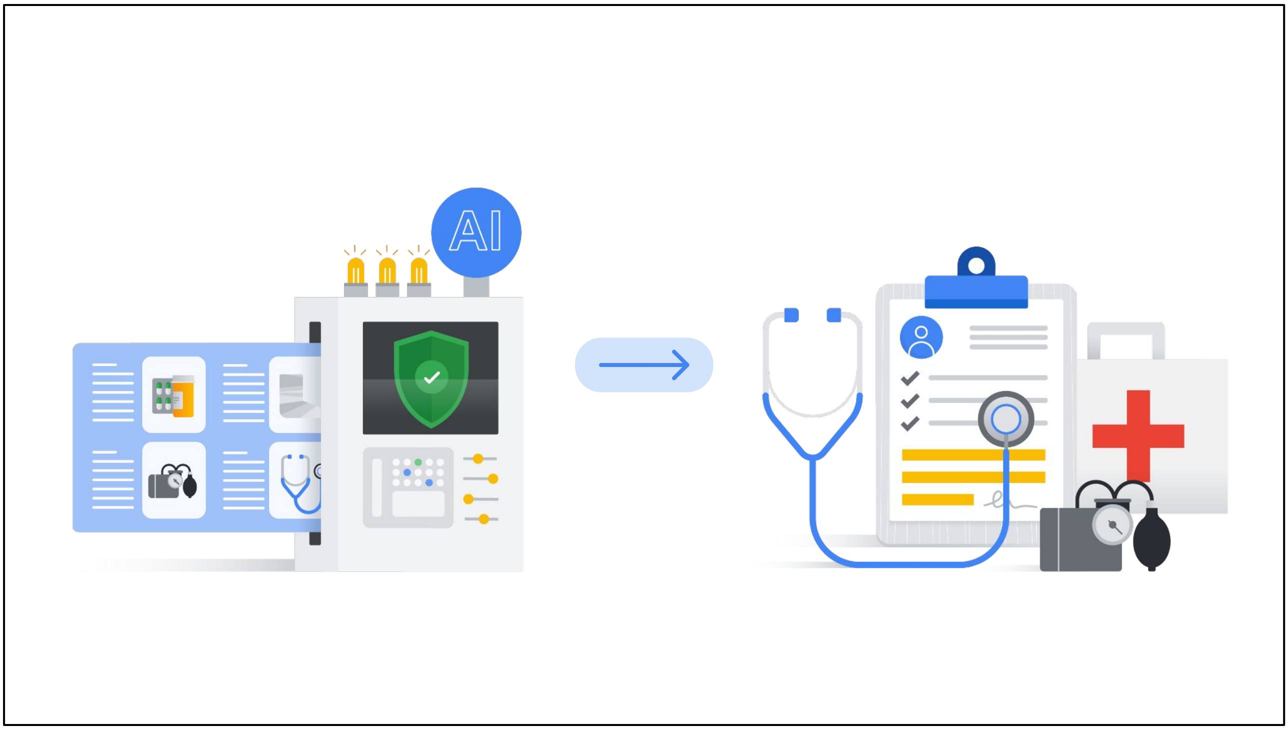
User Auth

Legal

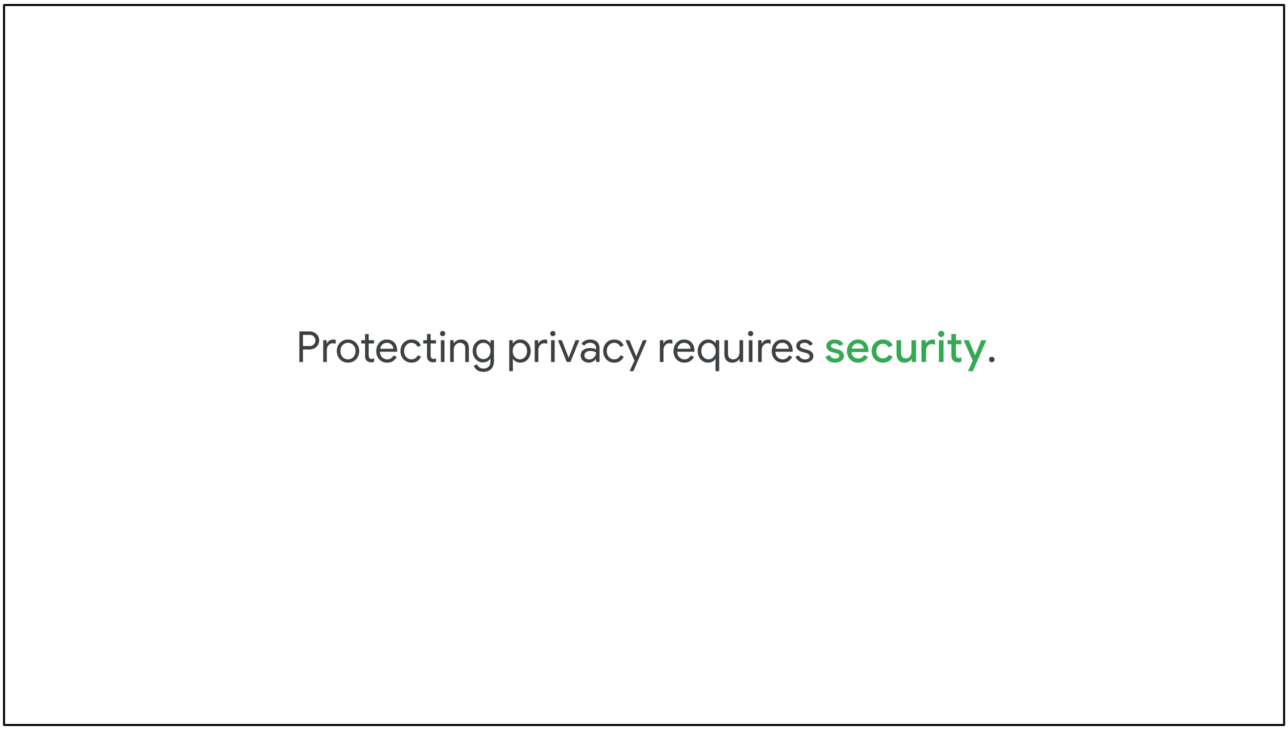
- Legal data, including intellectual property and trade secrets.



There may be enormous benefits to building a model that operates on sensitive data, such as an AI system trained on electronic health records to predict an individual's risk of chronic diseases,

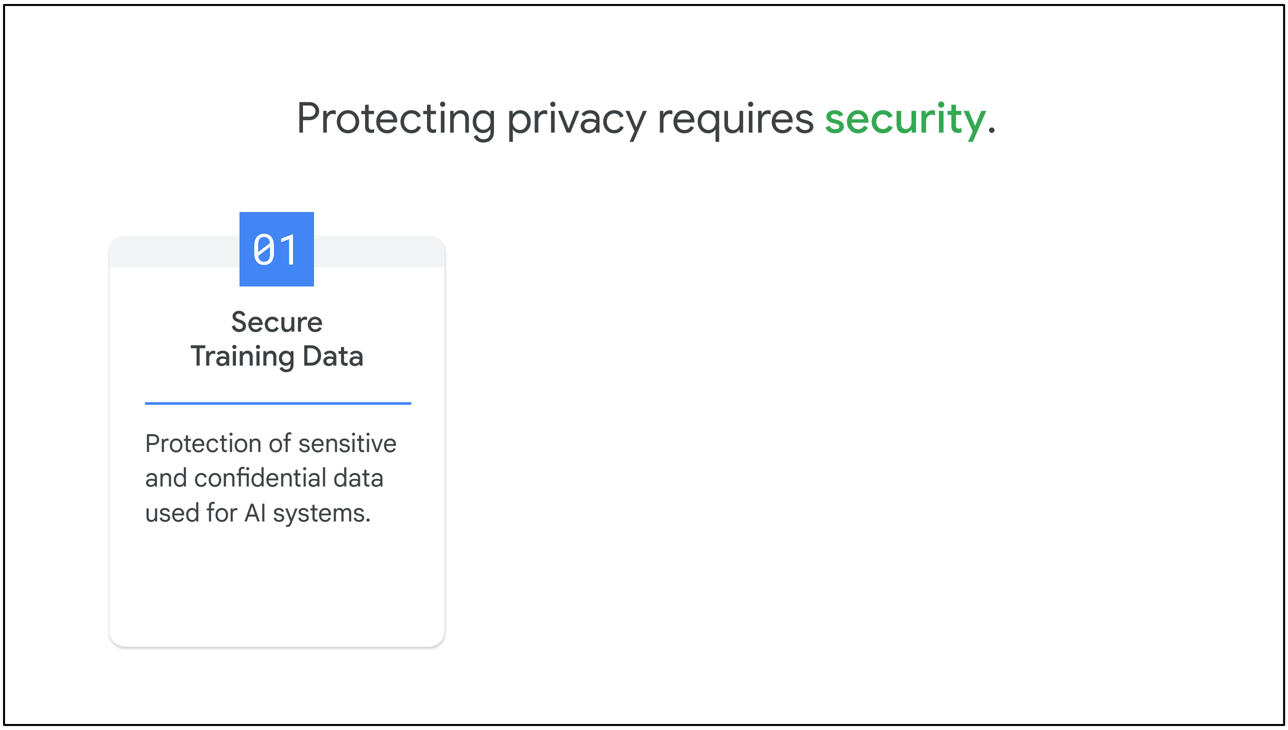


then deployed to make personalized health recommendations. However, it is always essential to consider the potential privacy implications in using sensitive data.



Protecting privacy requires security.

How can you ensure you are building a secure machine learning-based system? There are several aspects to consider.



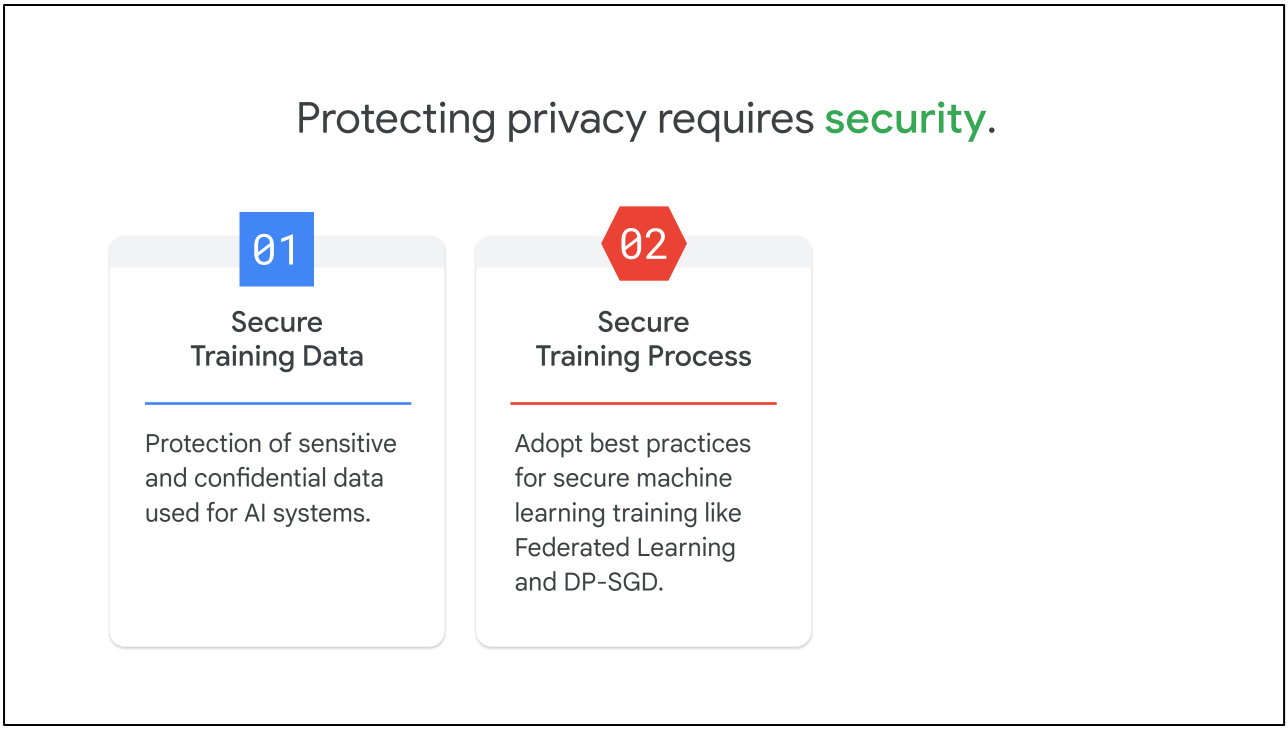
Protecting privacy requires security.

01

Secure Training Data

Protection of sensitive and confidential data used for AI systems.

Security in training data: Since machine learning is always trained on a dataset, the protection of sensitive and confidential data used for AI systems is critical.



Protecting privacy requires security.

01

Secure Training Data

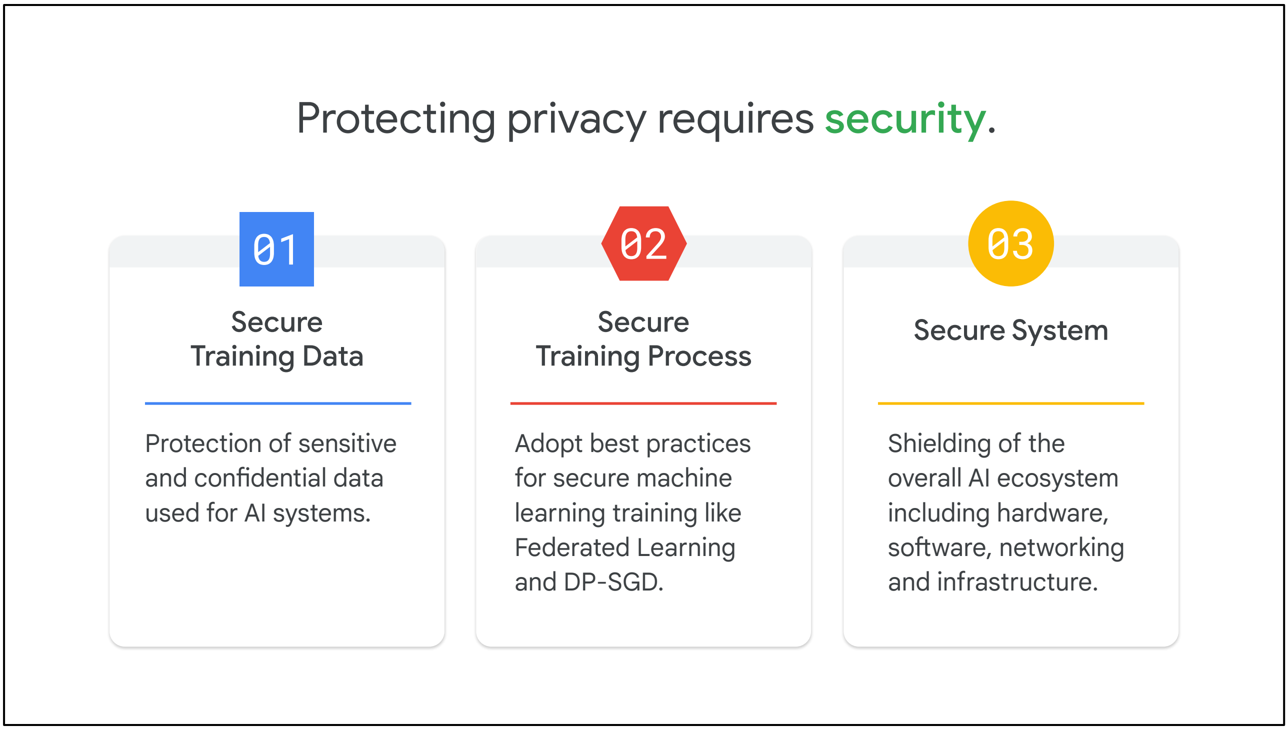
Protection of sensitive and confidential data used for AI systems.

02

Secure Training Process

Adopt best practices for secure machine learning training like Federated Learning and DP-SGD.

Security in training process: you can address security by adopting best practices for secure machine-learning training like Federated Learning and Differentially Private Stochastic Gradient Descent or DP-SGD.



Protecting privacy requires security.

01

Secure Training Data

Protection of sensitive and confidential data used for AI systems.

02

Secure Training Process

Adopt best practices for secure machine learning training like Federated Learning and DP-SGD.

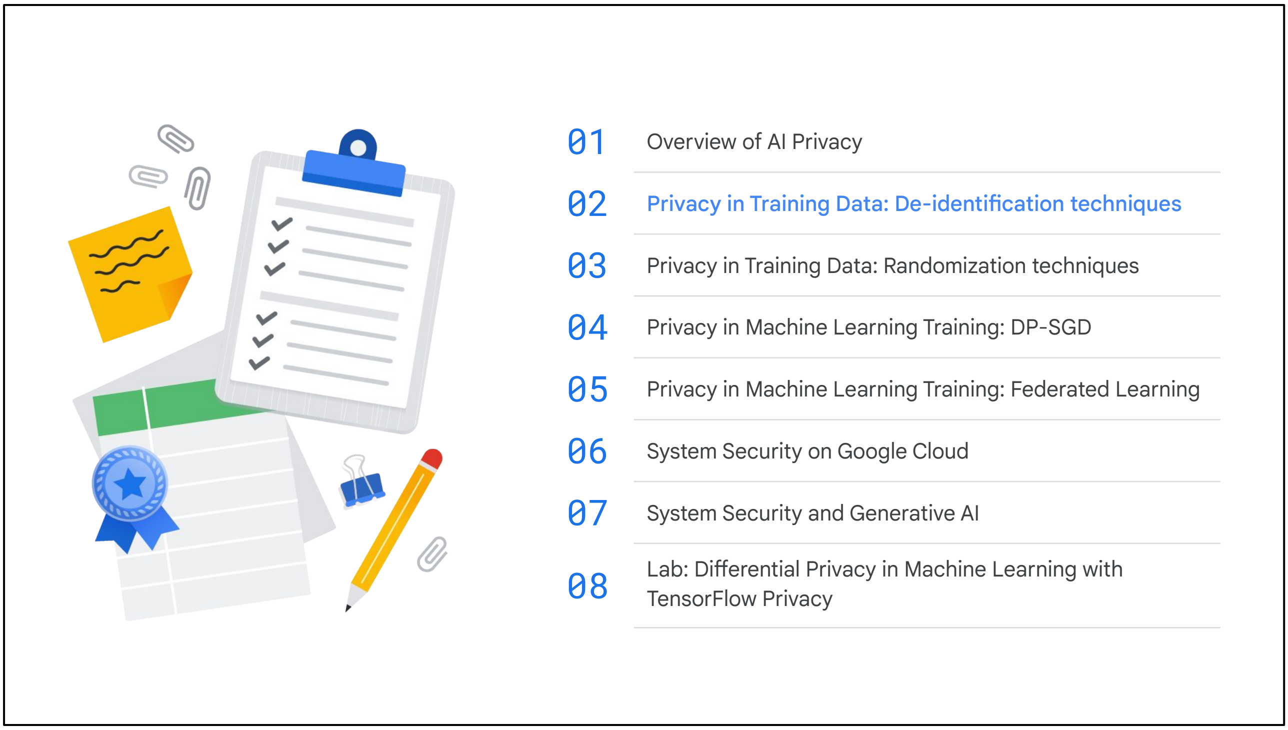
03

Secure System

Shielding of the overall AI ecosystem including hardware, software, networking and infrastructure.

Implementing privacy-preserving techniques across hardware, software, communication channels, and infrastructure is crucial for comprehensive AI system security.

We'll look at data security in detail first, and then discuss the other two aspects.



01 Overview of AI Privacy

02 Privacy in Training Data: De-identification techniques

03 Privacy in Training Data: Randomization techniques

04 Privacy in Machine Learning Training: DP-SGD

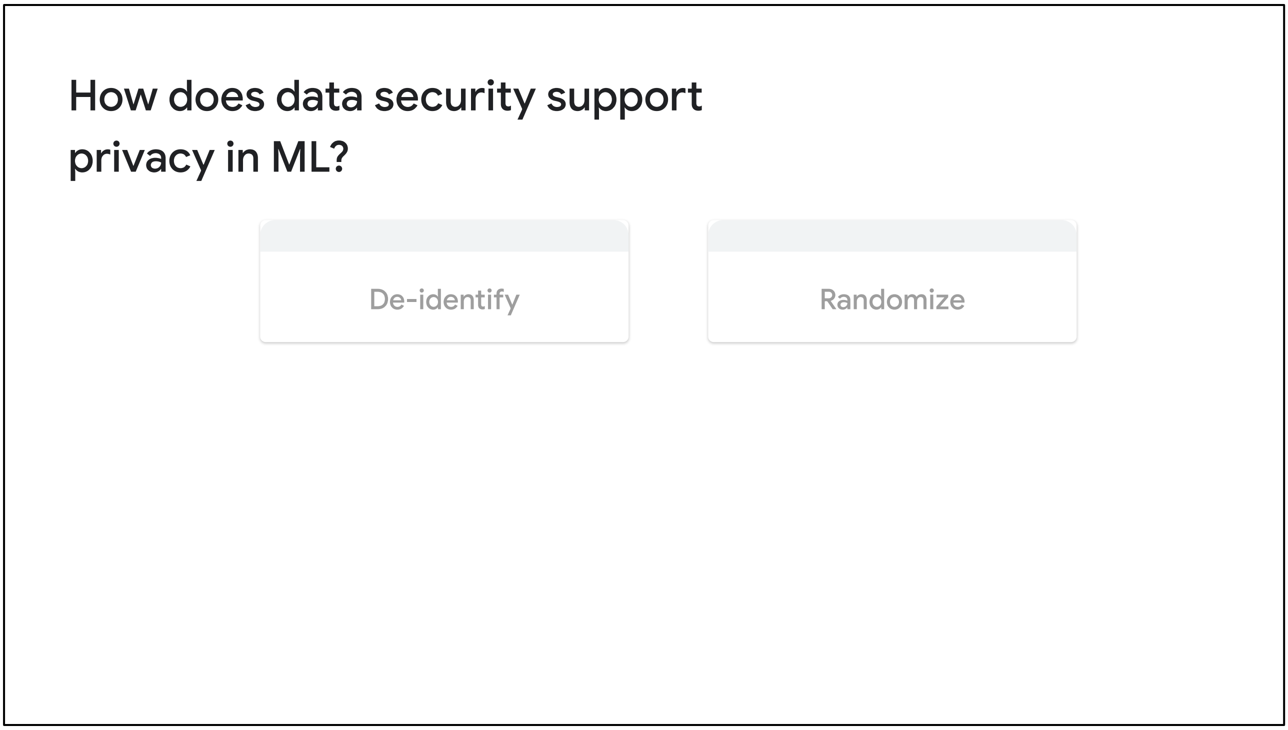
05 Privacy in Machine Learning Training: Federated Learning

06 System Security on Google Cloud

07 System Security and Generative AI

08 Lab: Differential Privacy in Machine Learning with TensorFlow Privacy

Let’s examine privacy in training data.



How does data security support privacy in ML?

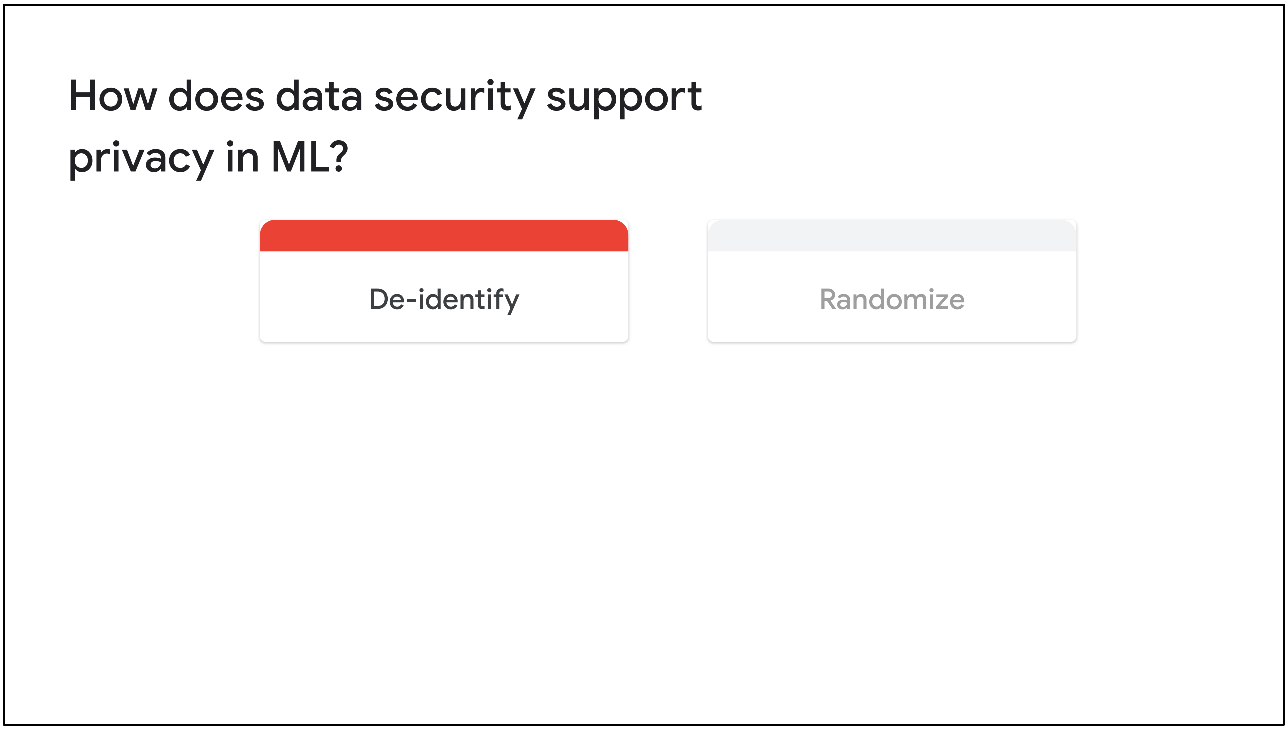
De-identify

Randomize

In our context of privacy in ML, data security is the protection of sensitive data used in AI systems.

Keep in mind that you want to minimize the use of sensitive data as much as possible. Preferably at the data collection phase, such as collecting aggregated data, rather than during data processing, or collecting individual interactions, for example.

We recommend two different, not exclusive, approaches to consider:

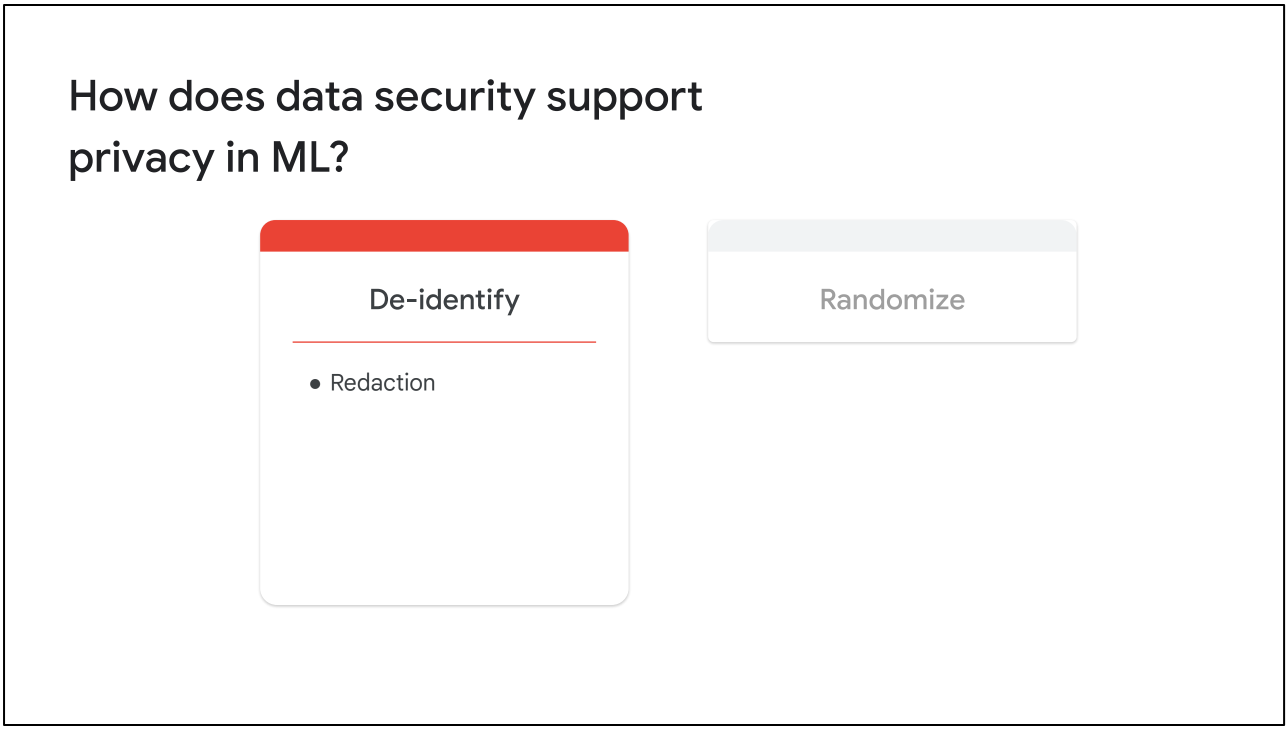


How does data security support privacy in ML?

De-identify

Randomize

1. De-identify your data through de-identification techniques such as:



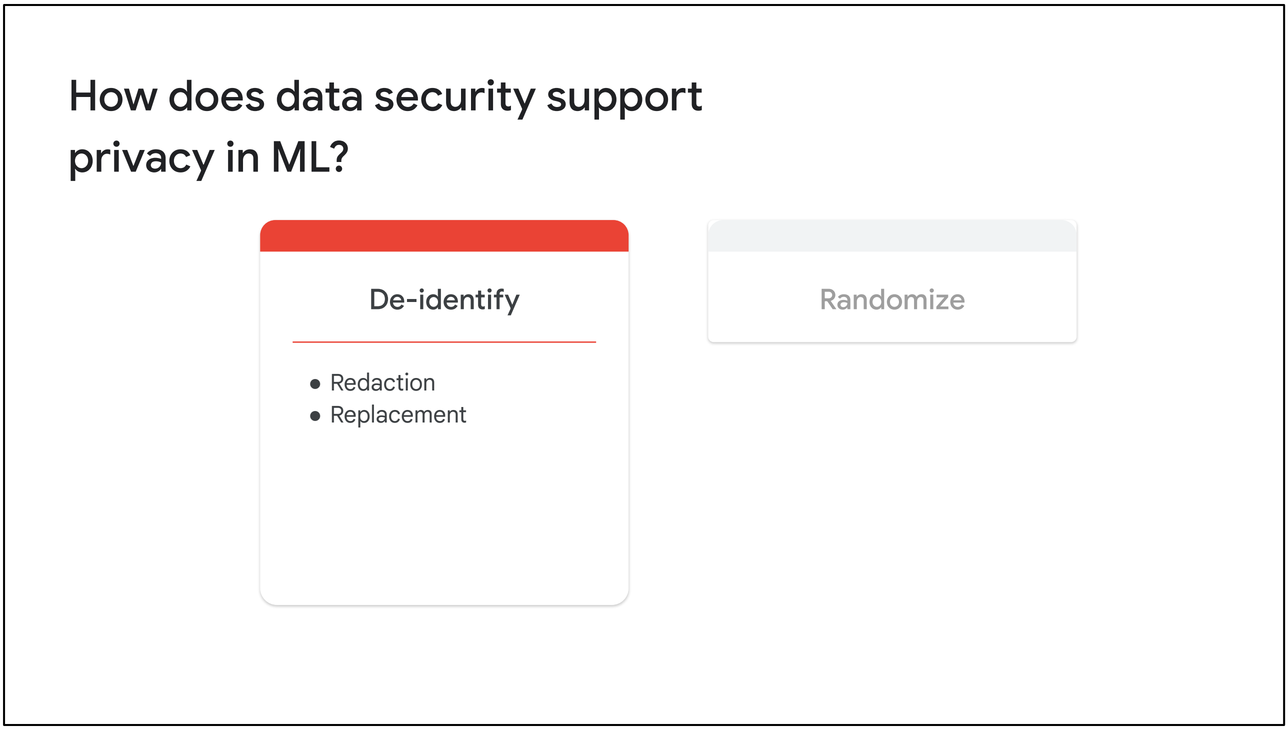
How does data security support privacy in ML?

De-identify

- Redaction

Randomize

redaction,



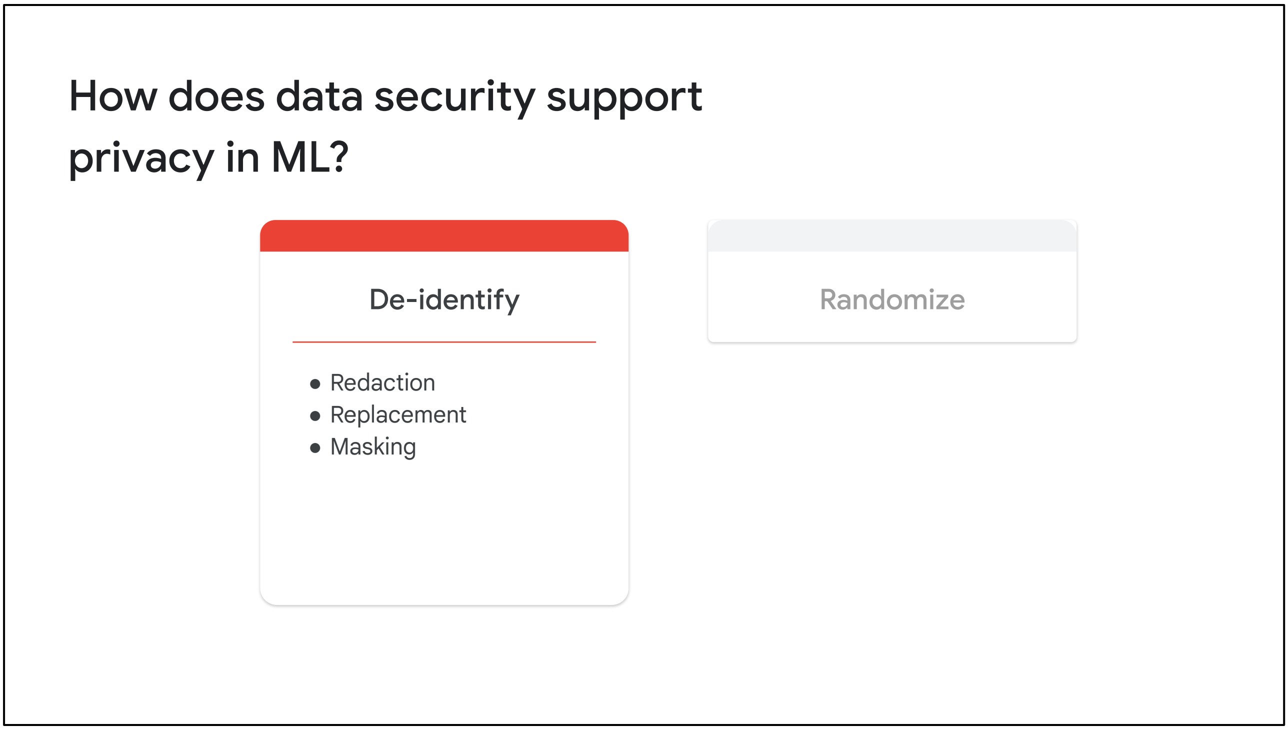
How does data security support privacy in ML?

De-identify

- Redaction
- Replacement

Randomize

replacement,



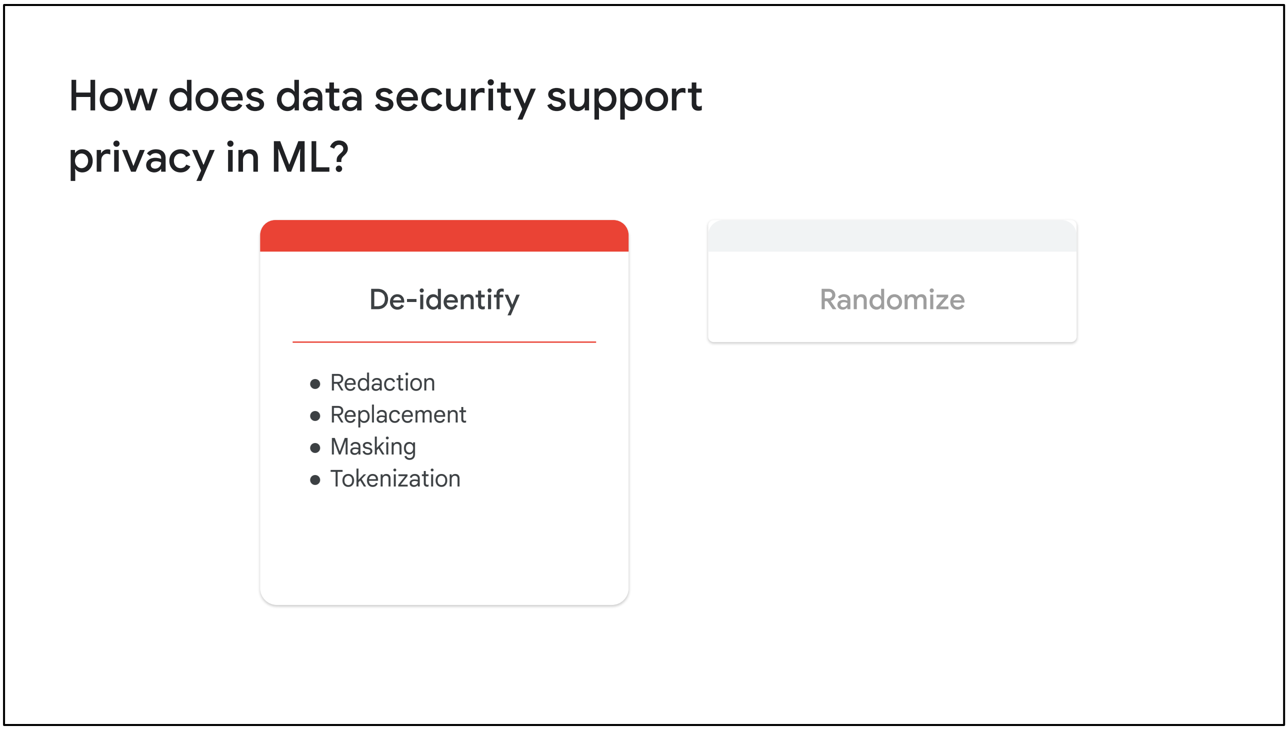
How does data security support privacy in ML?

De-identify

- Redaction
- Replacement
- Masking

Randomize

masking,



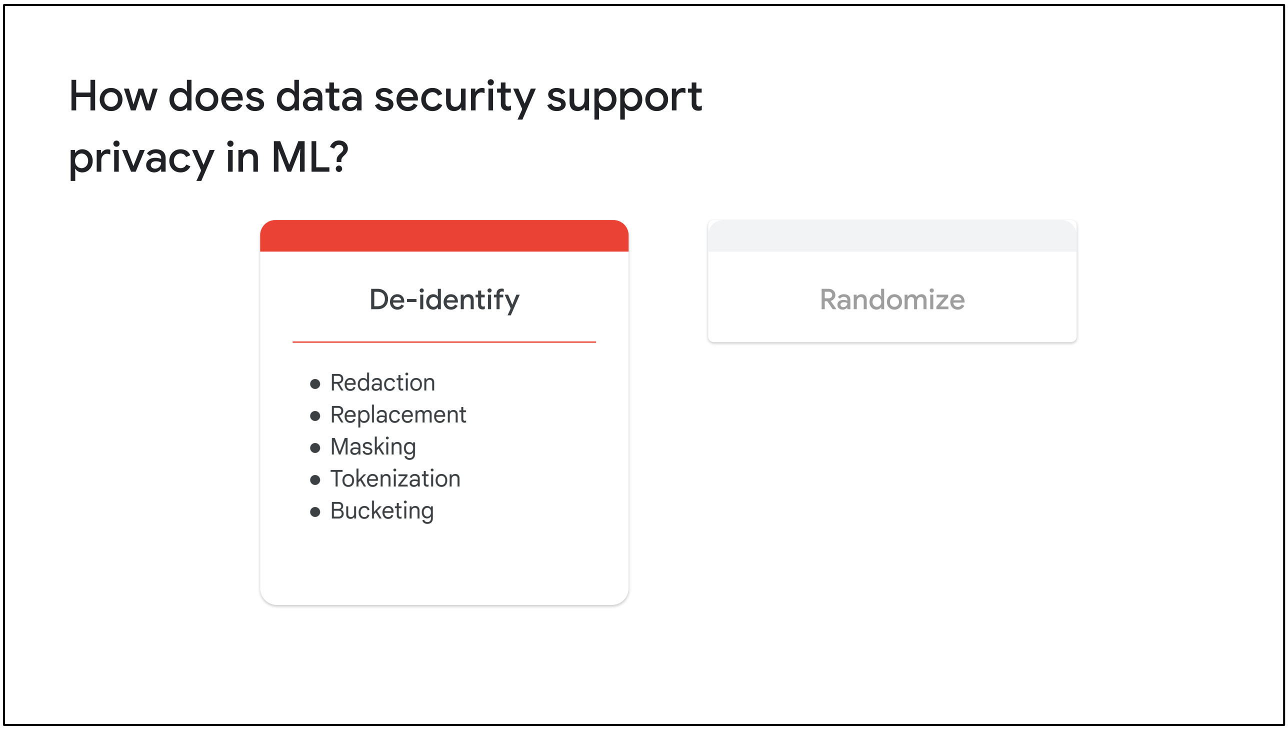
How does data security support privacy in ML?

De-identify

- Redaction
- Replacement
- Masking
- Tokenization

Randomize

tokenization,



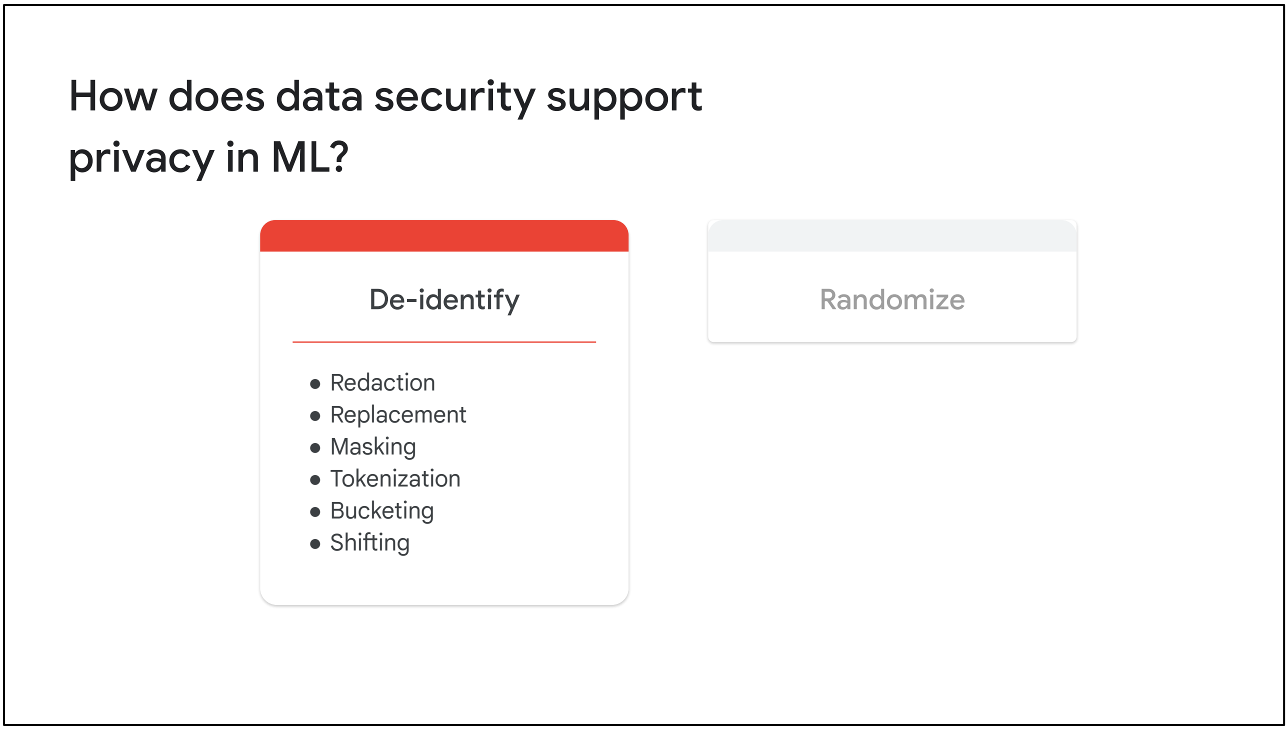
How does data security support privacy in ML?

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing

Randomize

bucketing



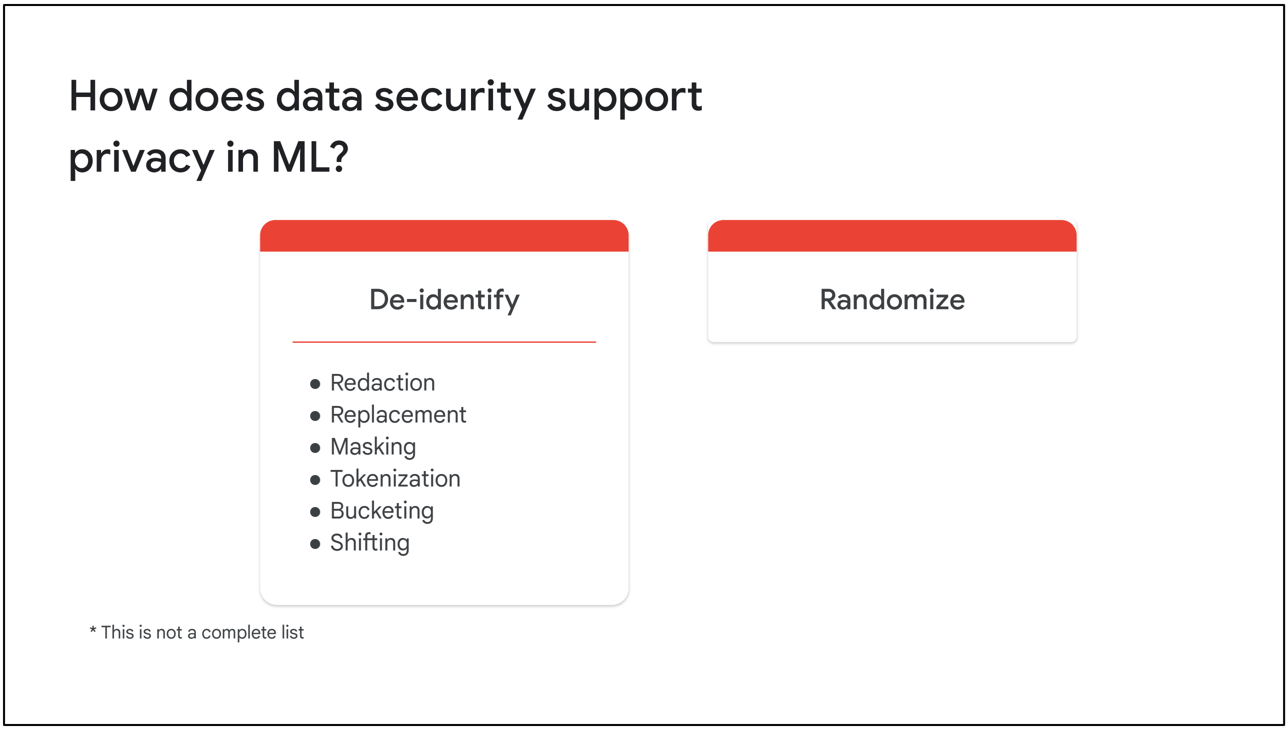
How does data security support privacy in ML?

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

and shifting.



How does data security support privacy in ML?

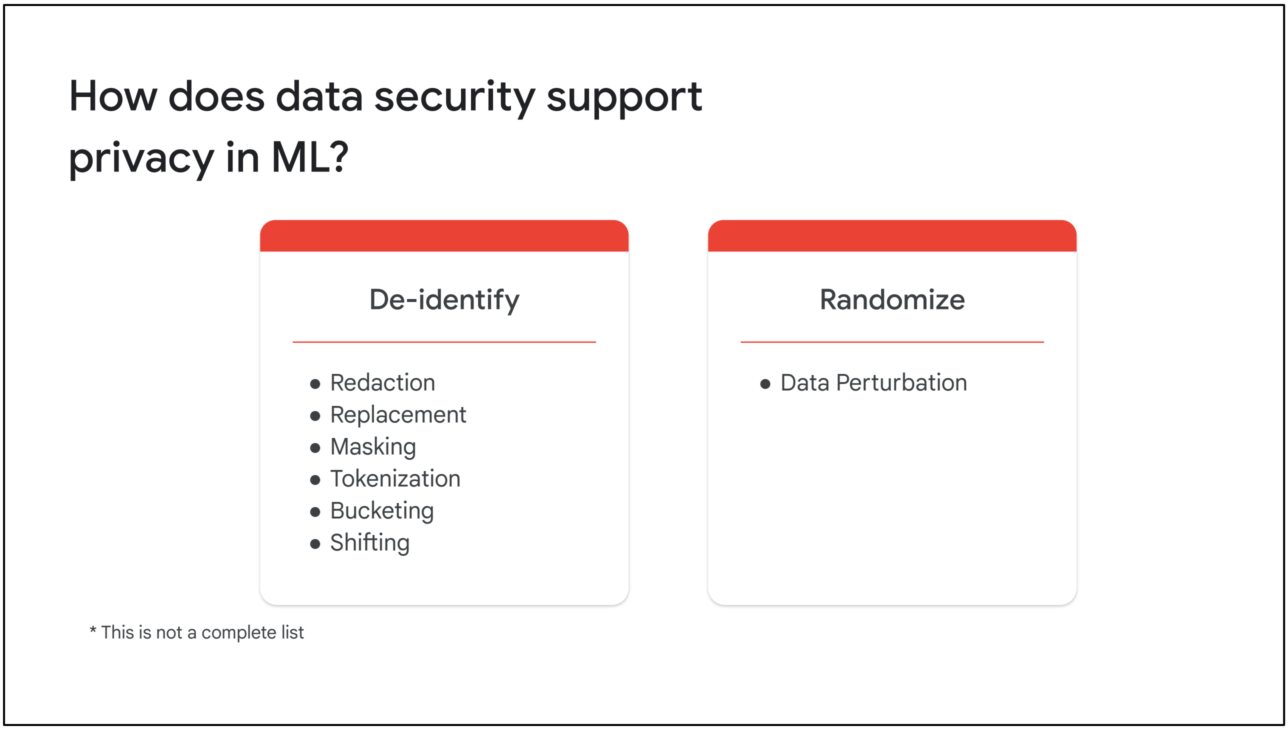
De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

* This is not a complete list

2. Randomize your data through methods such as



How does data security support privacy in ML?

De-identify

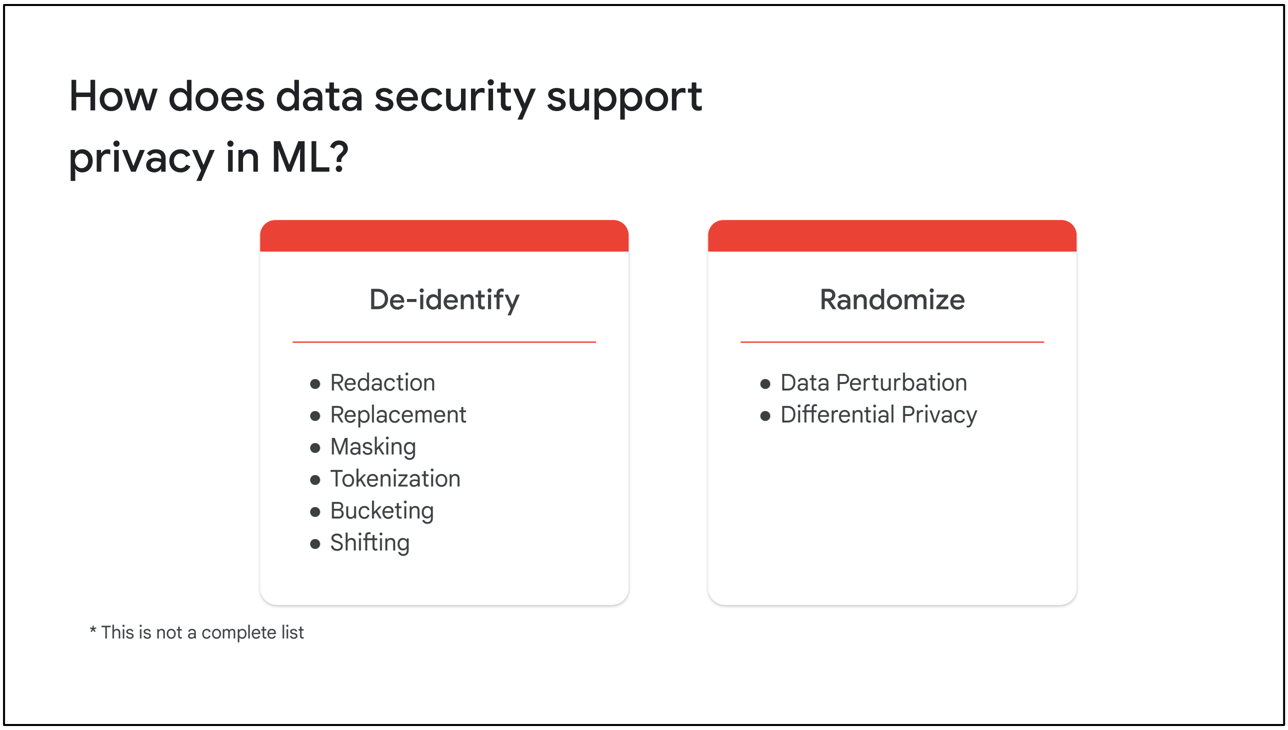
- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation

* This is not a complete list

data perturbation



How does data security support privacy in ML?

De-identify

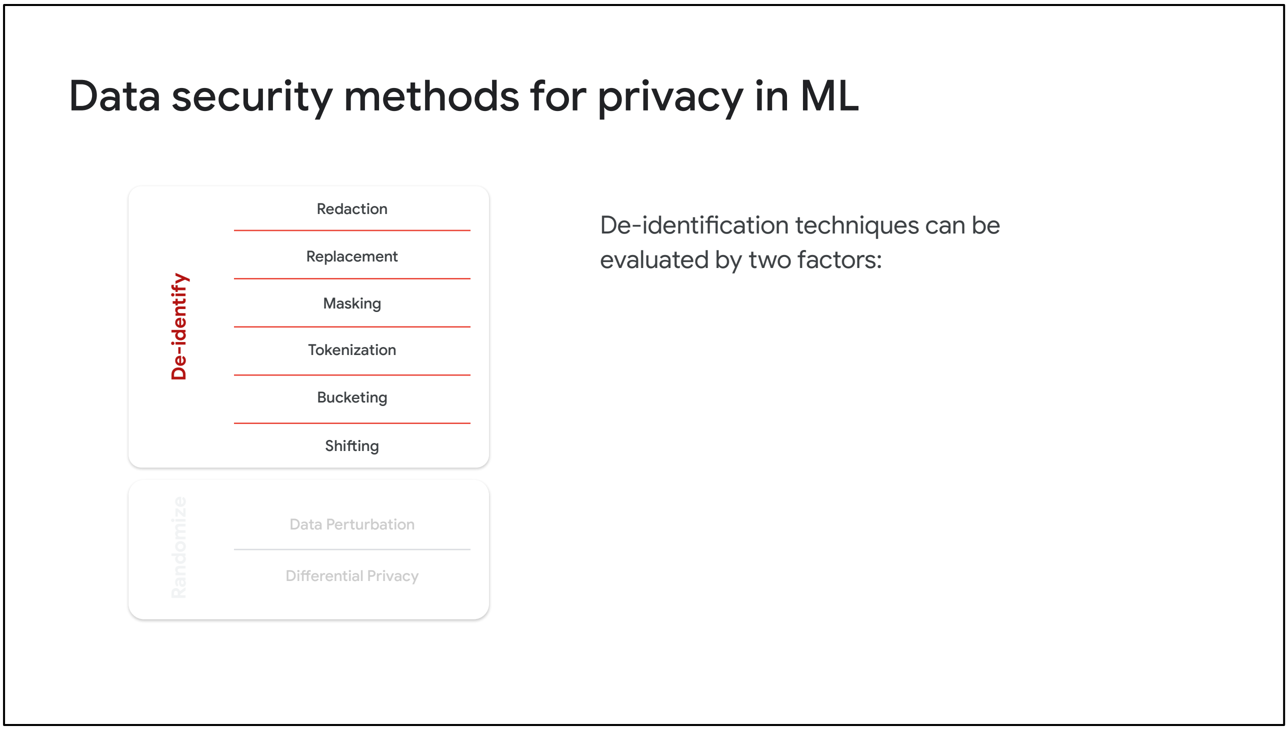
- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

* This is not a complete list

and differential privacy.



Data security methods for privacy in ML

De-identify

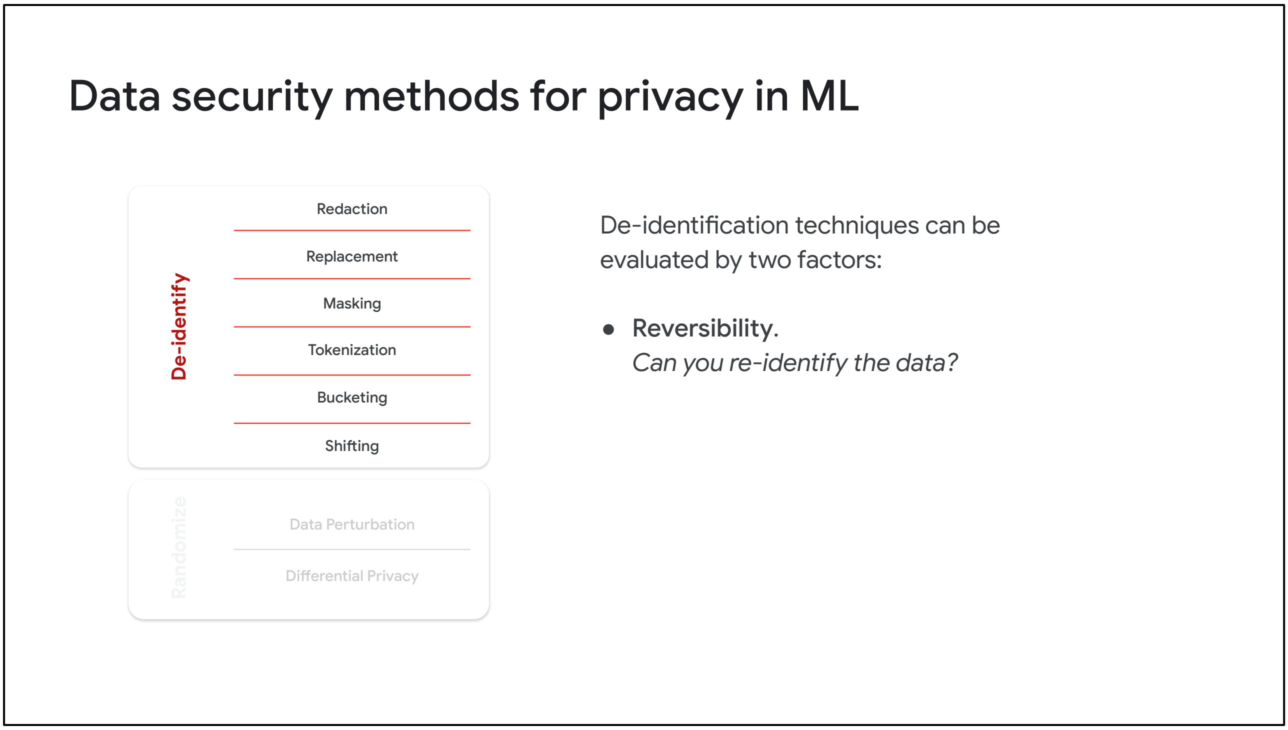
- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

De-identification techniques can be evaluated by two factors:

De-identification approaches can be evaluated by two factors:



Data security methods for privacy in ML

De-identify

Redaction

Replacement

Masking

Tokenization

Bucketing

Shifting

Randomize

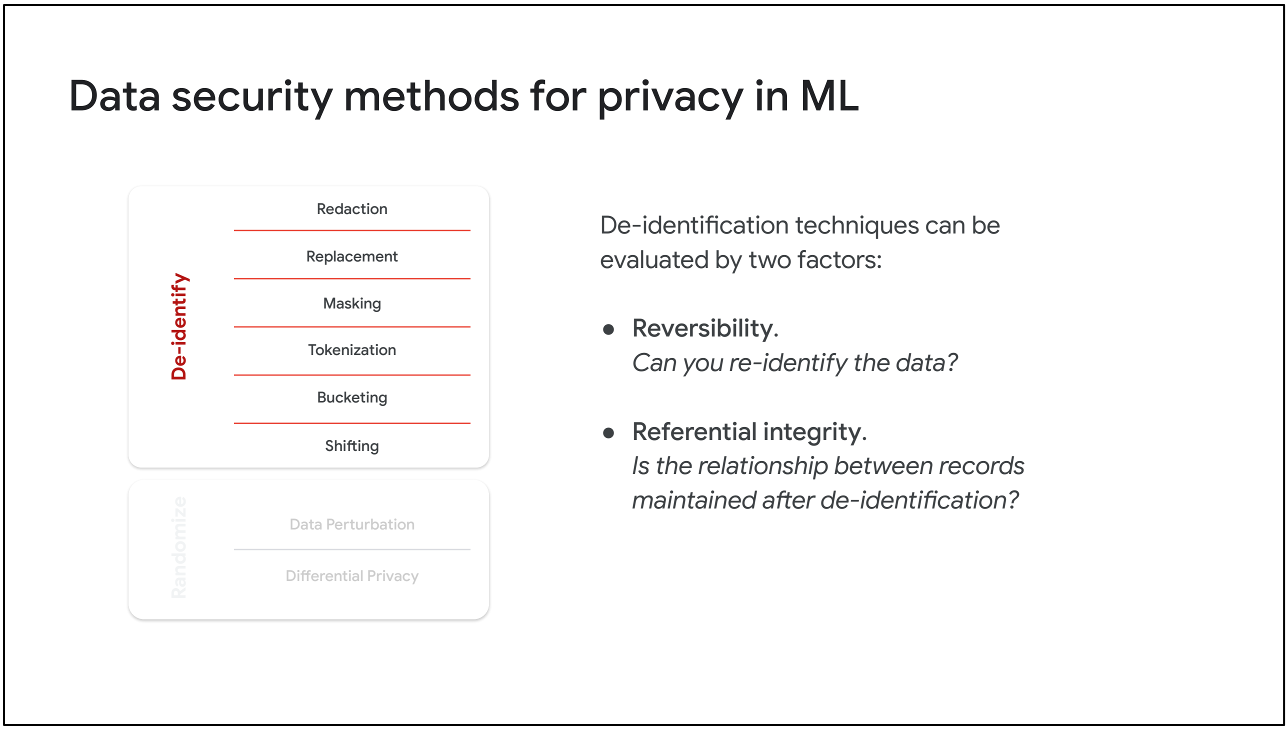
Data Perturbation

Differential Privacy

De-identification techniques can be evaluated by two factors:

- **Reversibility.**
  Can you re-identify the data?

a. Reversibility: If an approach is reversible, can you re-identify the data?



Data security methods for privacy in ML

De-identify

Redaction

Replacement

Masking

Tokenization

Bucketing

Shifting

Randomize

Data Perturbation

Differential Privacy

De-identification techniques can be evaluated by two factors:

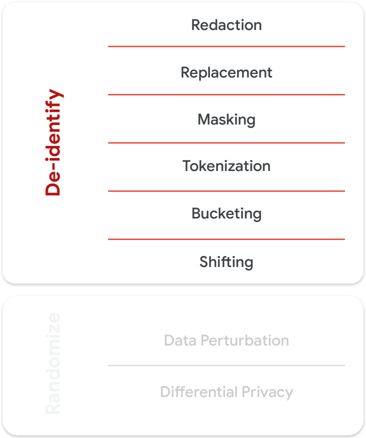
- **Reversibility.**
  Can you re-identify the data?
- **Referential integrity.**
  Is the relationship between records maintained after de-identification?

b. Referential integrity: Is the relationship between records maintained after de-identification?

Whether these factors are a positive aspect or a potential risk depends on the specific use case and data privacy requirements.

For example, reversibility is great for auditability and recovery, but it can easily lead to data leakage, otherwise known as re-identification.
The same risks exist with referential integrity (although, less likely). But referential integrity has the added benefits of improving ML performance and consistency when guaranteed

Data security methods for privacy in ML



De-identify

Redaction

Replacement

Masking

Tokenization

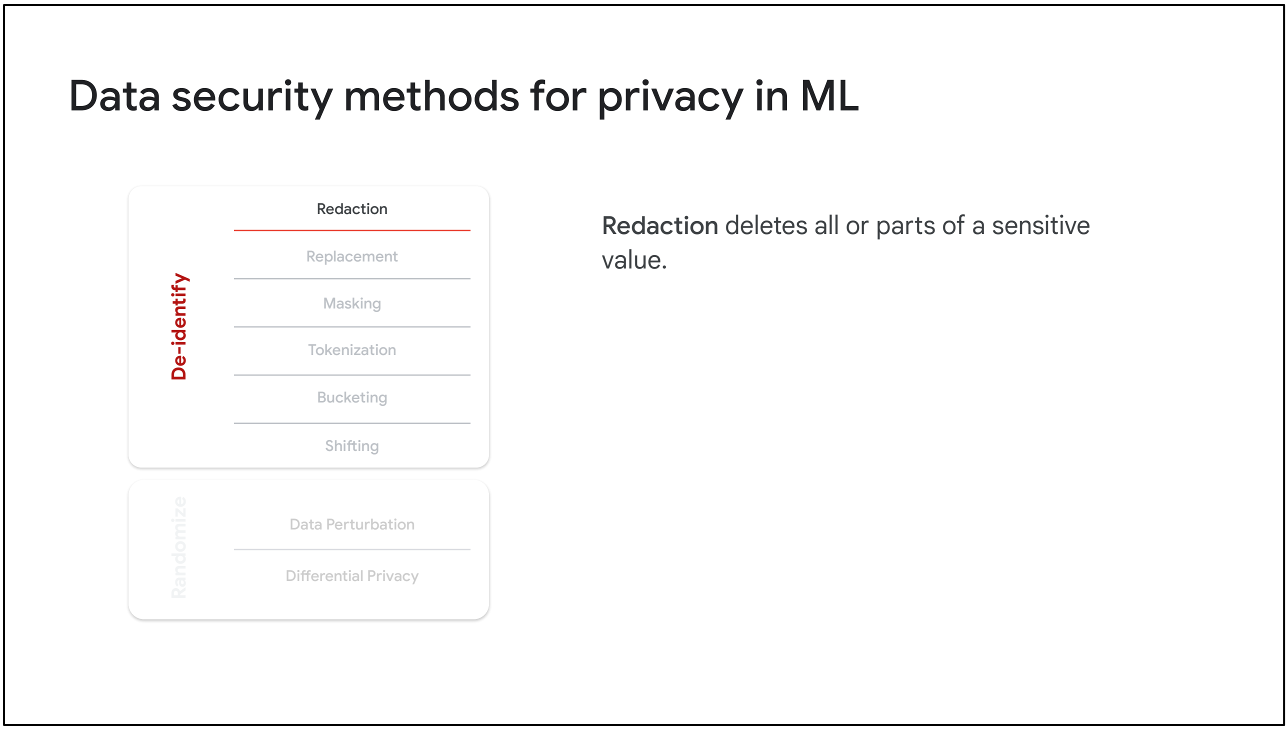
Bucketing

Shifting

Randomize

Data Perturbation

Differential Privacy



Data security methods for privacy in ML

De-identify

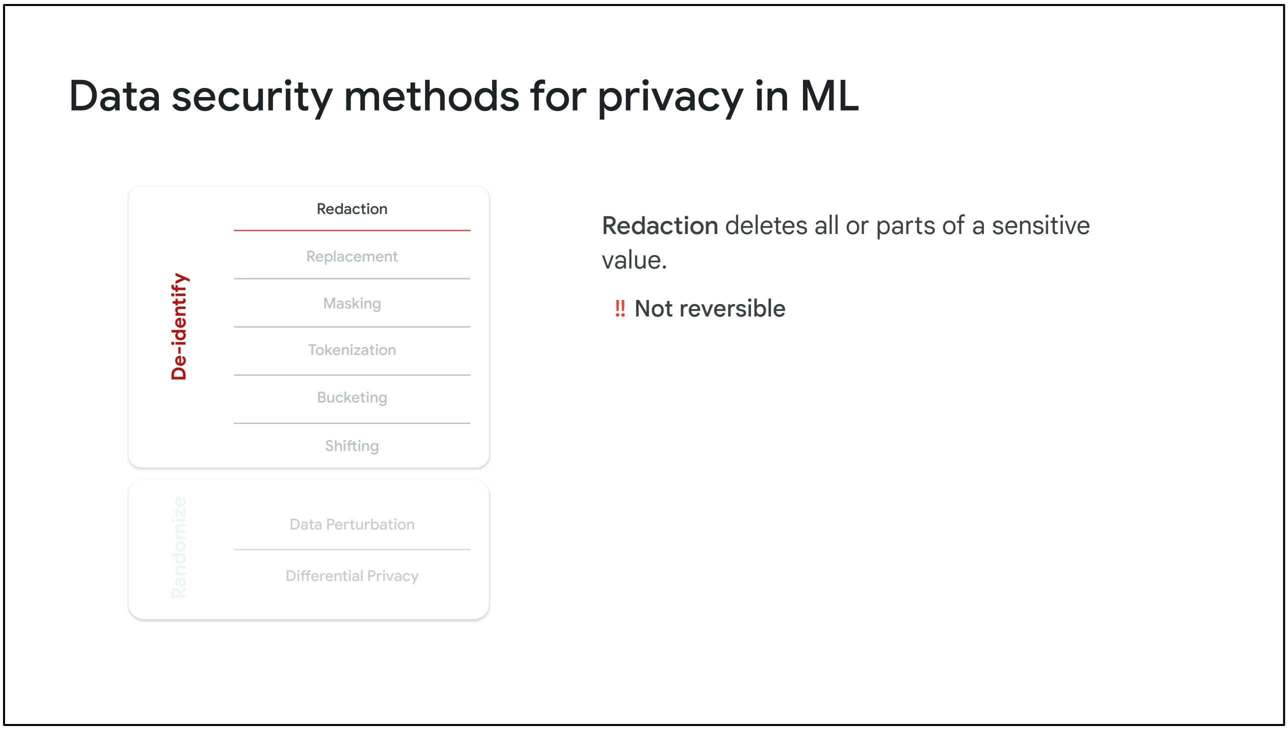
- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Redaction deletes all or parts of a sensitive value.

First de-identification technique: redaction.
Redaction is the process of deleting all or parts of a sensitive value.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

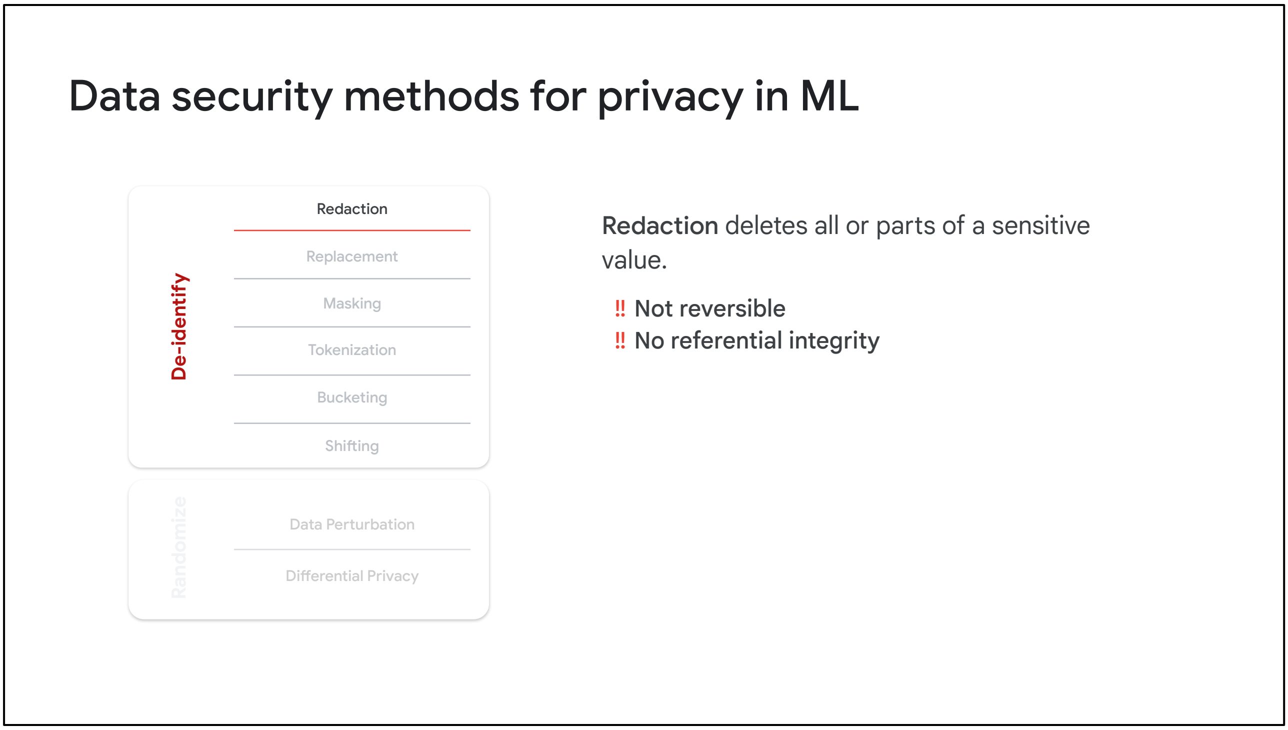
Randomize

- Data Perturbation
- Differential Privacy

Redaction deletes all or parts of a sensitive value.

‼ Not reversible

This approach is not reversible,



Data security methods for privacy in ML

De-identify

Redaction

Replacement

Masking

Tokenization

Bucketing

Shifting

Randomize

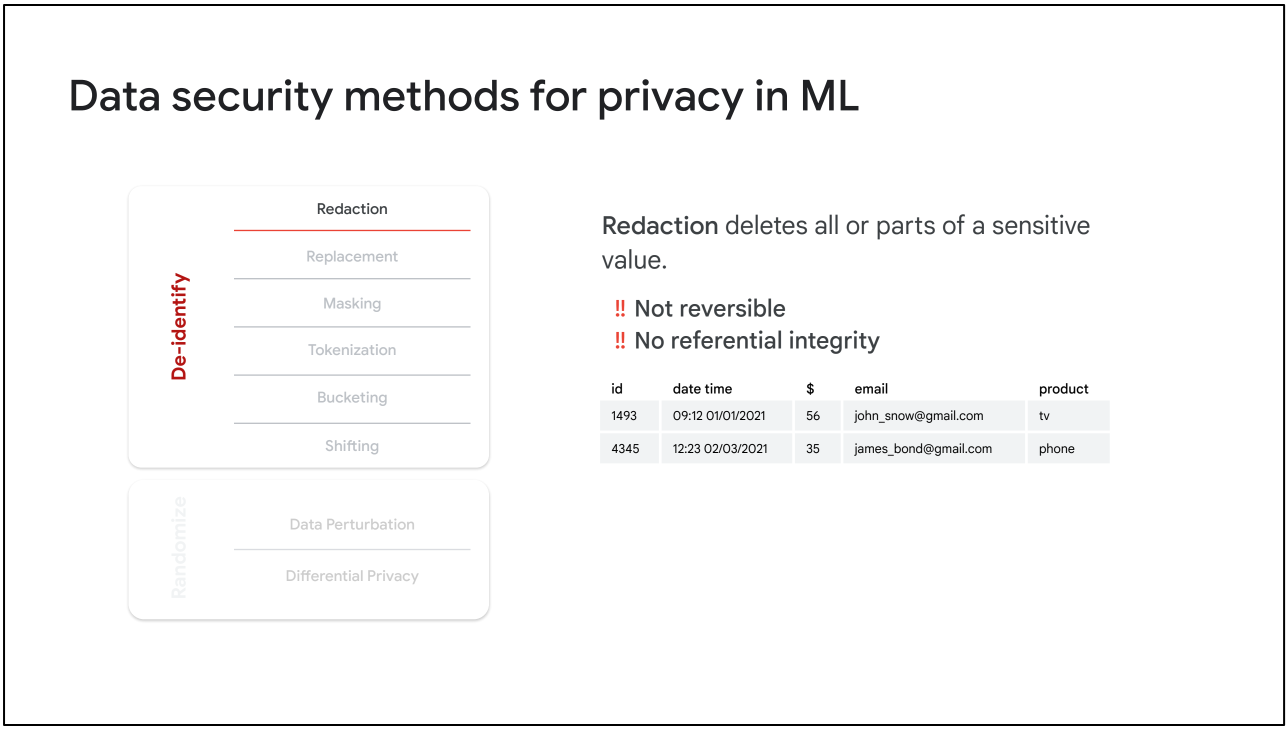
Data Perturbation

Differential Privacy

Redaction deletes all or parts of a sensitive value.

‼ Not reversible
‼ No referential integrity

and it does not maintain referential integrity.



Data security methods for privacy in ML

De-identify

Redaction

Replacement

Masking

Tokenization

Bucketing

Shifting

Randomize

Data Perturbation

Differential Privacy

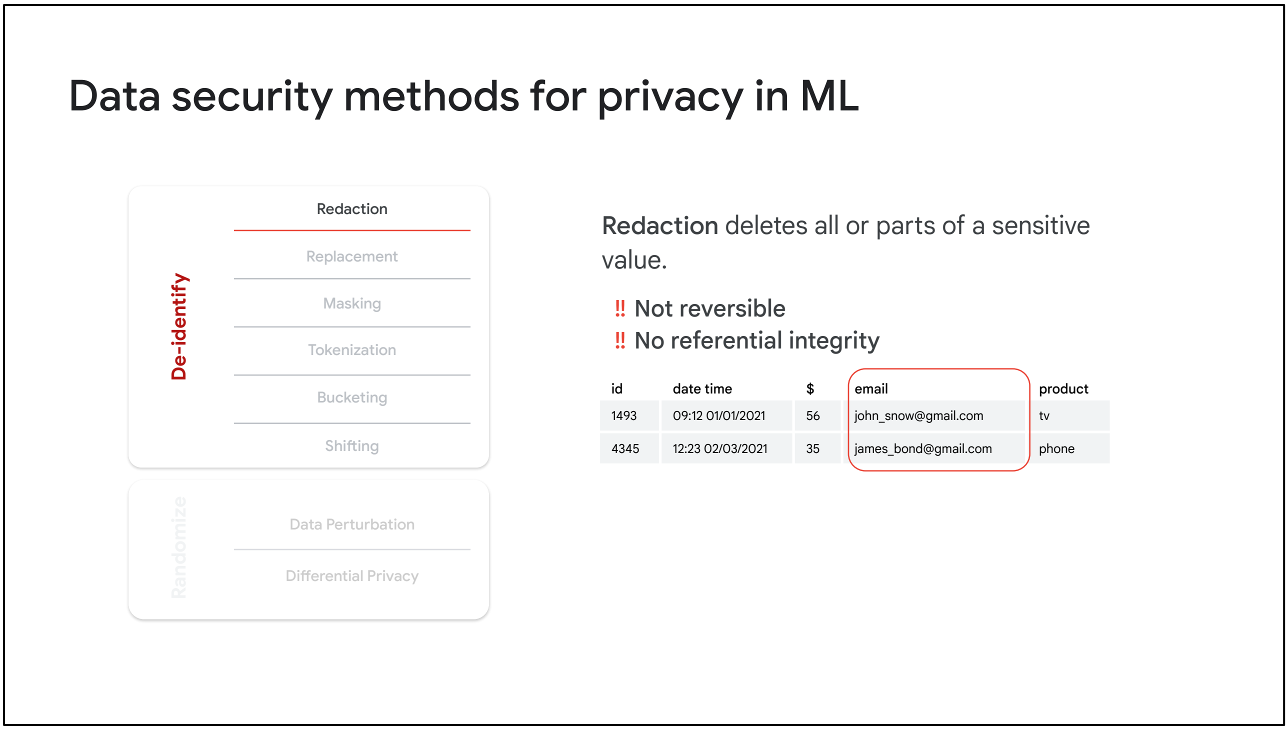
Redaction deletes all or parts of a sensitive value.

‼ Not reversible

‼ No referential integrity

id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

Let’s look at an example. Let’s say you have a transactional table with id, datetime, amount in dollars, email, and product. We will follow this example for all the de-identification techniques.



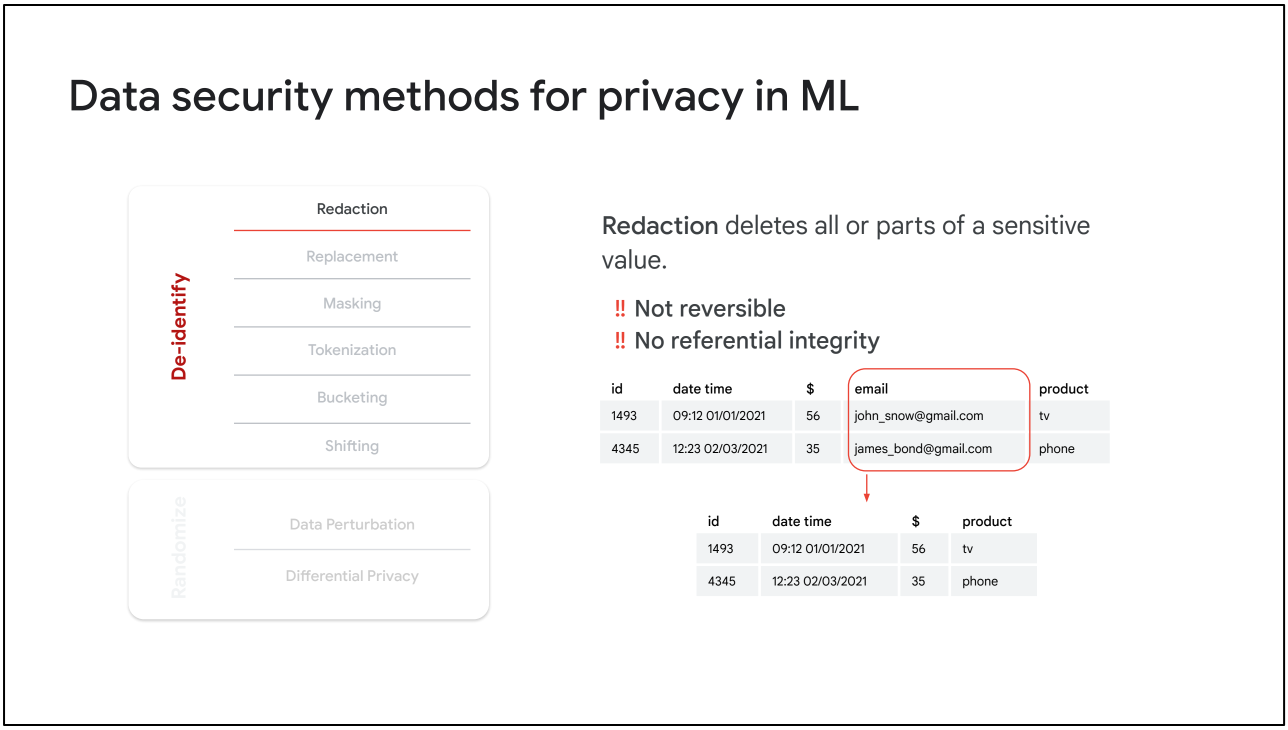
Data security methods for privacy in ML

Redaction deletes all or parts of a sensitive value.

‼ Not reversible
‼ No referential integrity

id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

If you want to redact the column “email”, then you can simply delete it from the view.



Data security methods for privacy in ML

De-identify

- **Redaction**
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Redaction deletes all or parts of a sensitive value.

‼ Not reversible

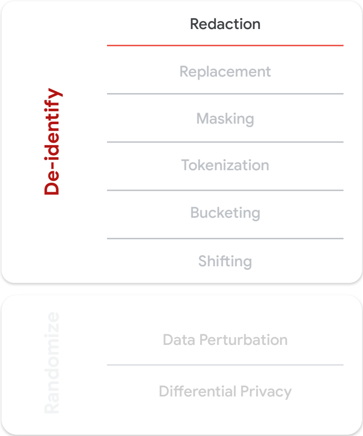
‼ No referential integrity

id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

id	date time	$	product
1493	09:12 01/01/2021	56	tv
4345	12:23 02/03/2021	35	phone

Redaction removes data, which can lead to a loss of information that could be valuable for machine learning models. This can affect the accuracy, generalizability, and overall effectiveness of the model.
For example, if we redact locations from a dataset, this action might hinder the model's ability to learn geospatial patterns.

Data security methods for privacy in ML



De-identify

Redaction

Replacement

Masking

Tokenization

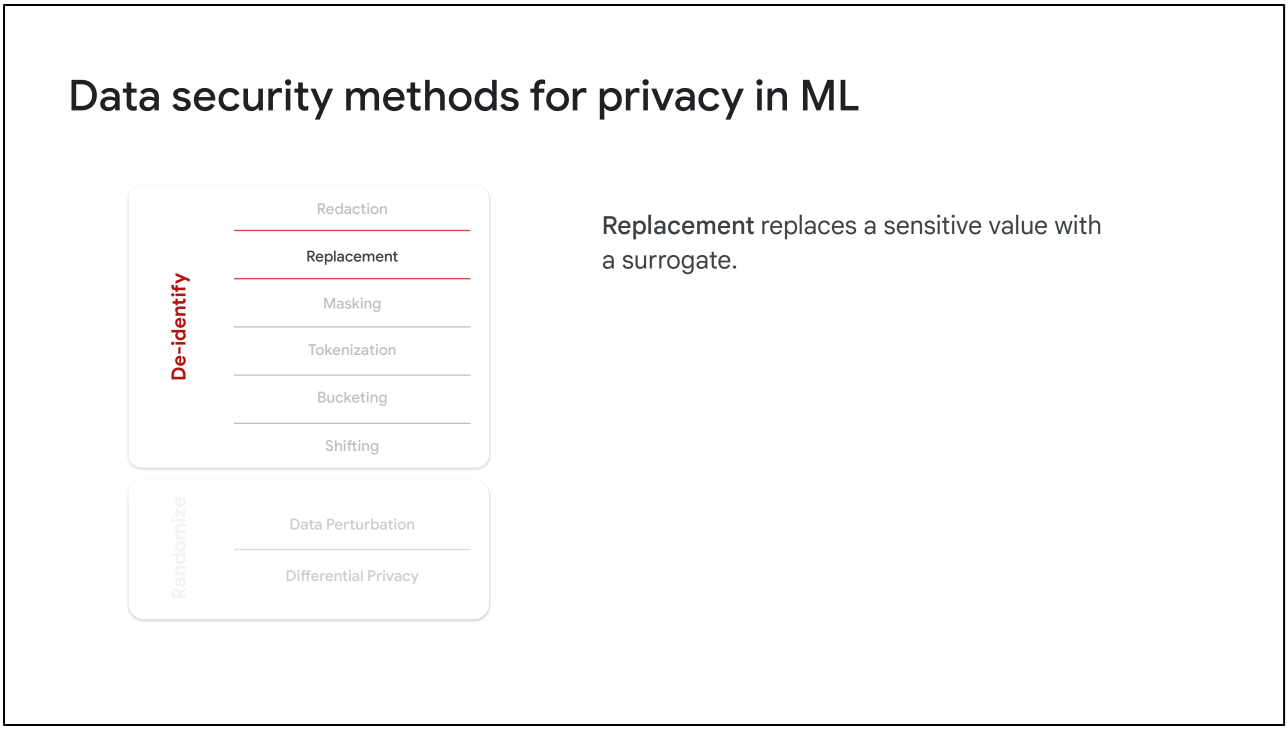
Bucketing

Shifting

Randomize

Data Perturbation

Differential Privacy



Data security methods for privacy in ML

De-identify

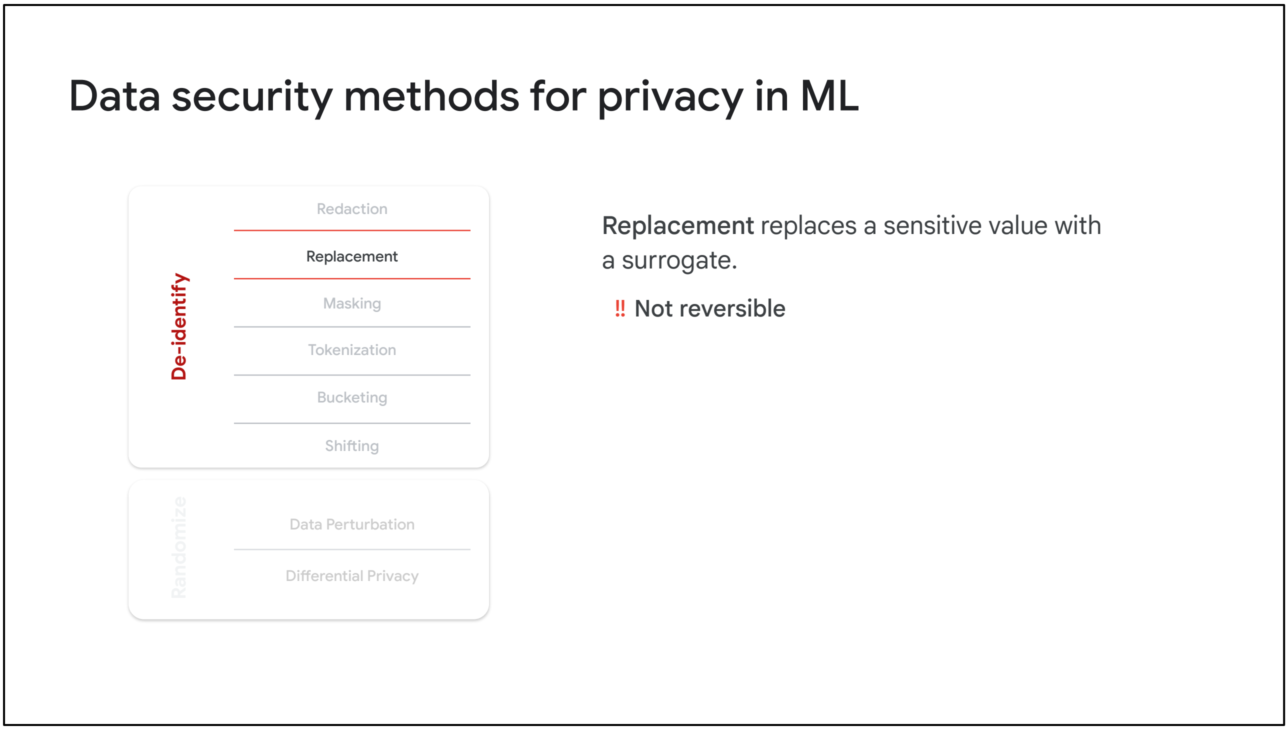
- Redaction
- **Replacement**
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Replacement replaces a sensitive value with a surrogate.

The second de-identification technique is replacement. This means replacing a sensitive value with a surrogate value that you specify.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

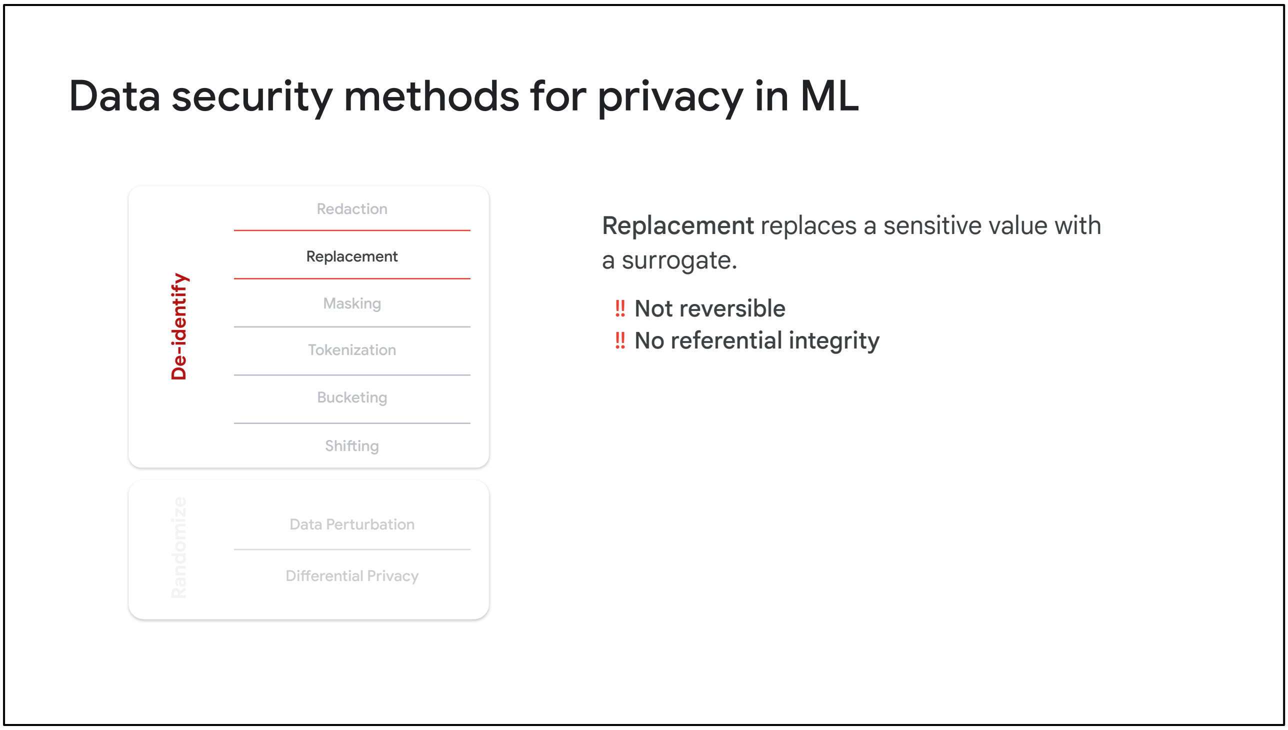
Randomize

- Data Perturbation
- Differential Privacy

Replacement replaces a sensitive value with a surrogate.

‼ Not reversible

It can be only one value, or a value randomly selected from a list of values.
The approach is irreversible,



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

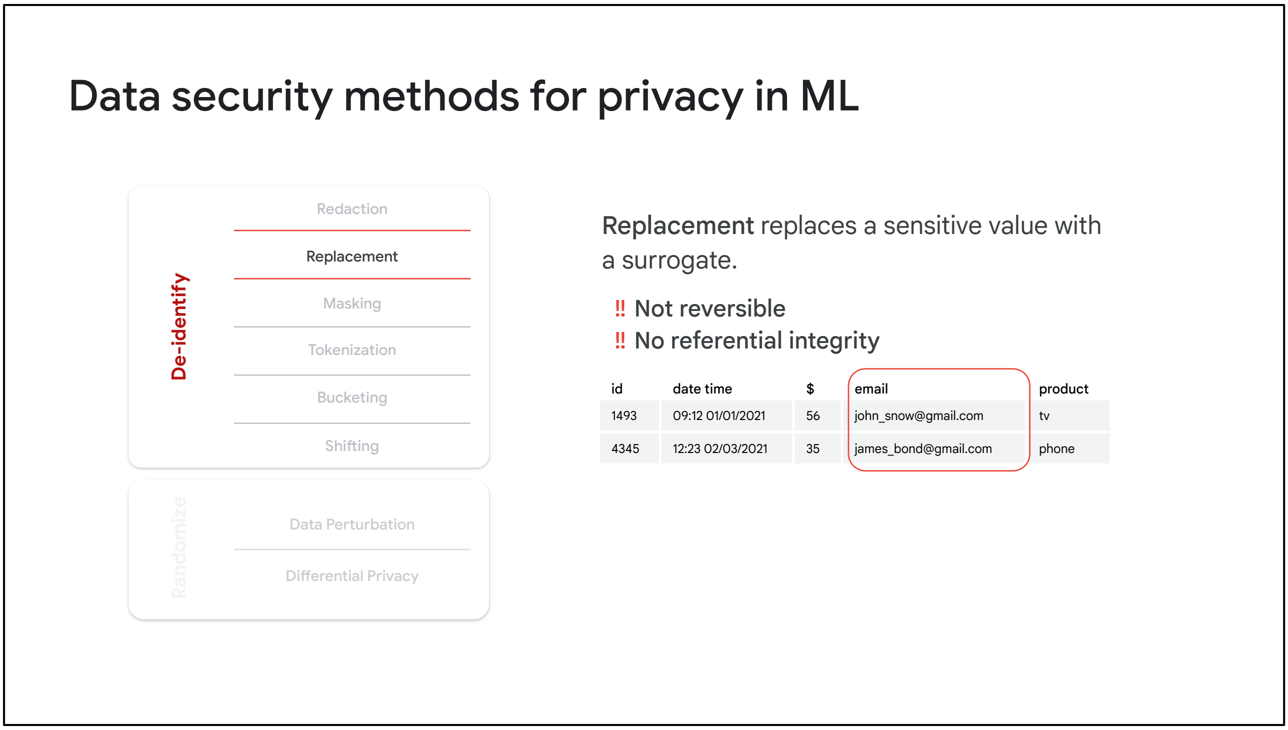
Randomize

- Data Perturbation
- Differential Privacy

Replacement replaces a sensitive value with a surrogate.

‼ Not reversible
‼ No referential integrity

and it does not maintain referential integrity. Note that replacement can have the same negative effect as redaction does on the model's ability to learn due to data loss.



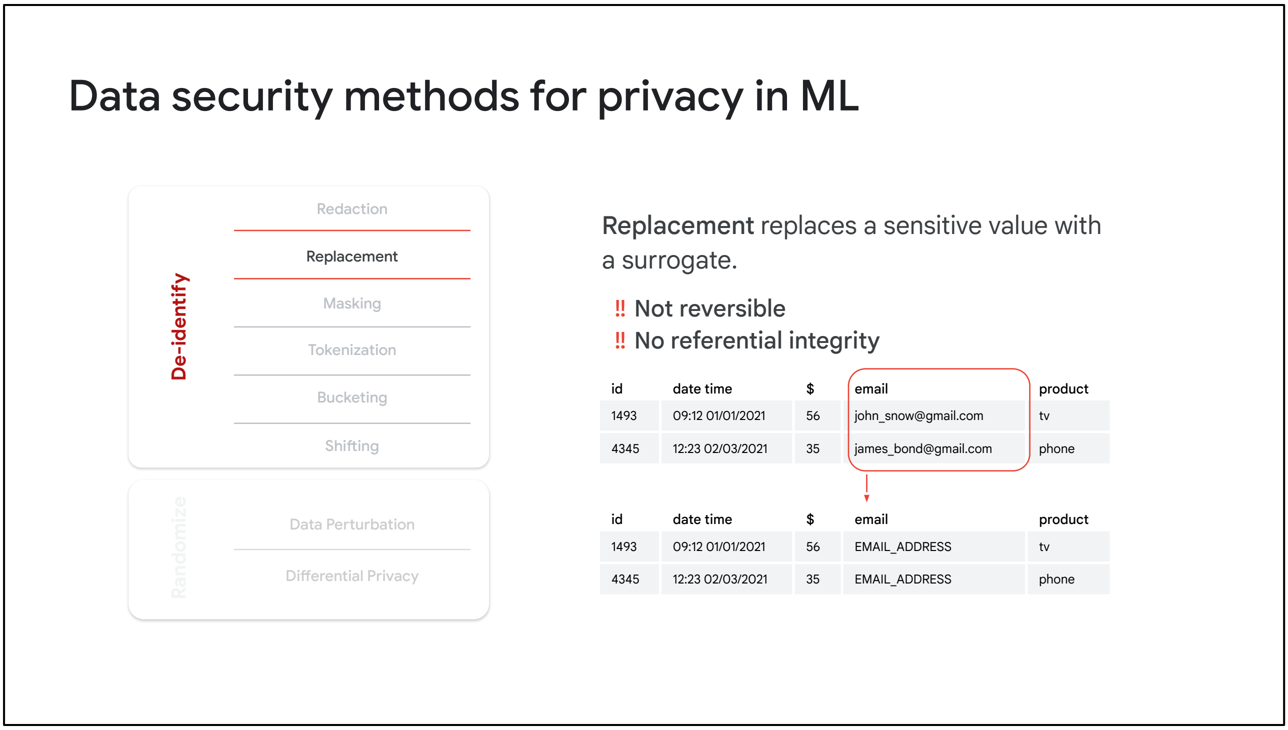
Data security methods for privacy in ML

Replacement replaces a sensitive value with a surrogate.

‼ Not reversible
‼ No referential integrity

id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

For example, let’s say you want to apply for a replacement on the email column.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Replacement replaces a sensitive value with a surrogate.

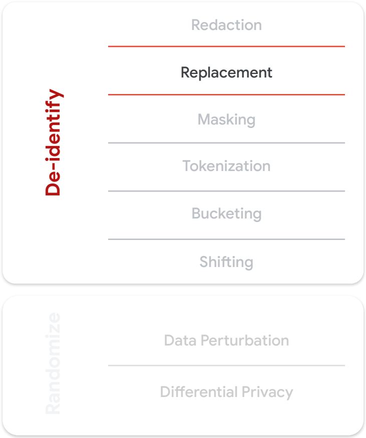
‼ Not reversible
‼ No referential integrity

id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

id	date time	$	email	product
1493	09:12 01/01/2021	56	EMAIL_ADDRESS	tv
4345	12:23 02/03/2021	35	EMAIL_ADDRESS	phone

Then you can replace all emails with its infotype, i.e. EMAIL_ADDRESS.
It’s worth mentioning that, in this example, it has the same effect as redaction. Thus, we need to be aware of the same limitations here about data loss.

Data security methods for privacy in ML

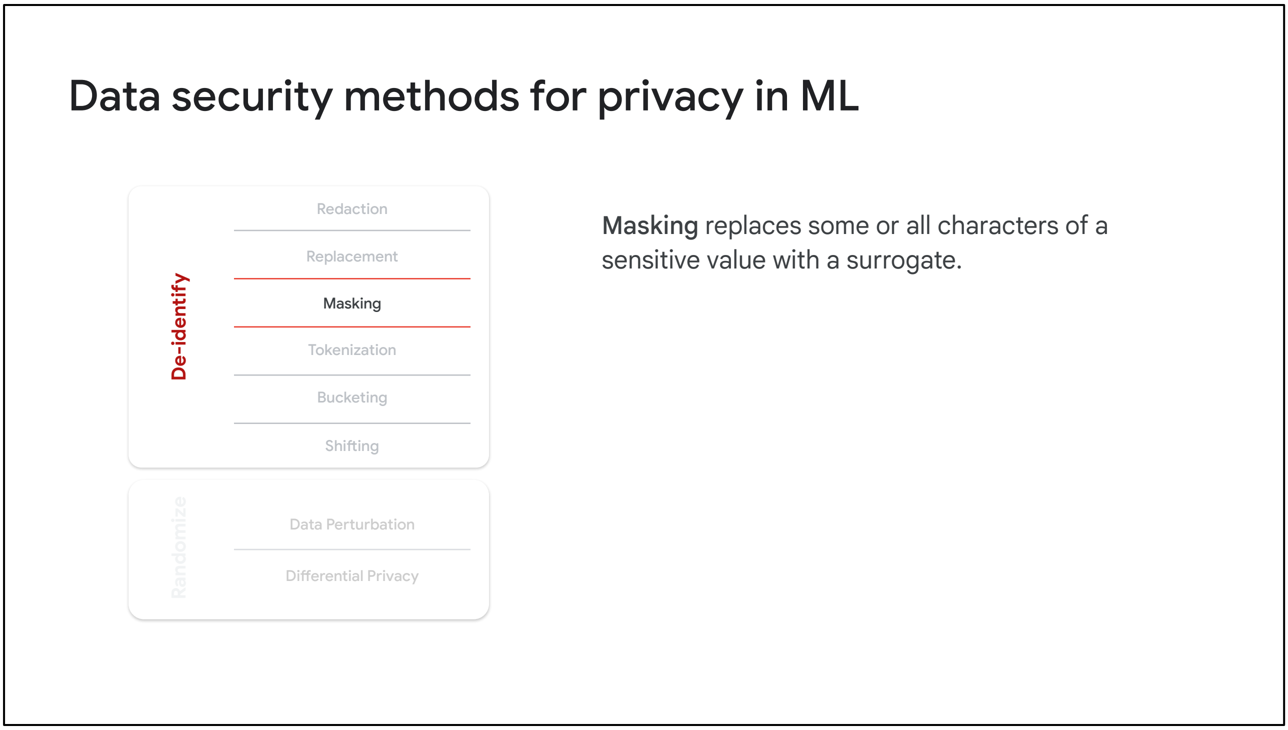


De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy



Data security methods for privacy in ML

De-identify

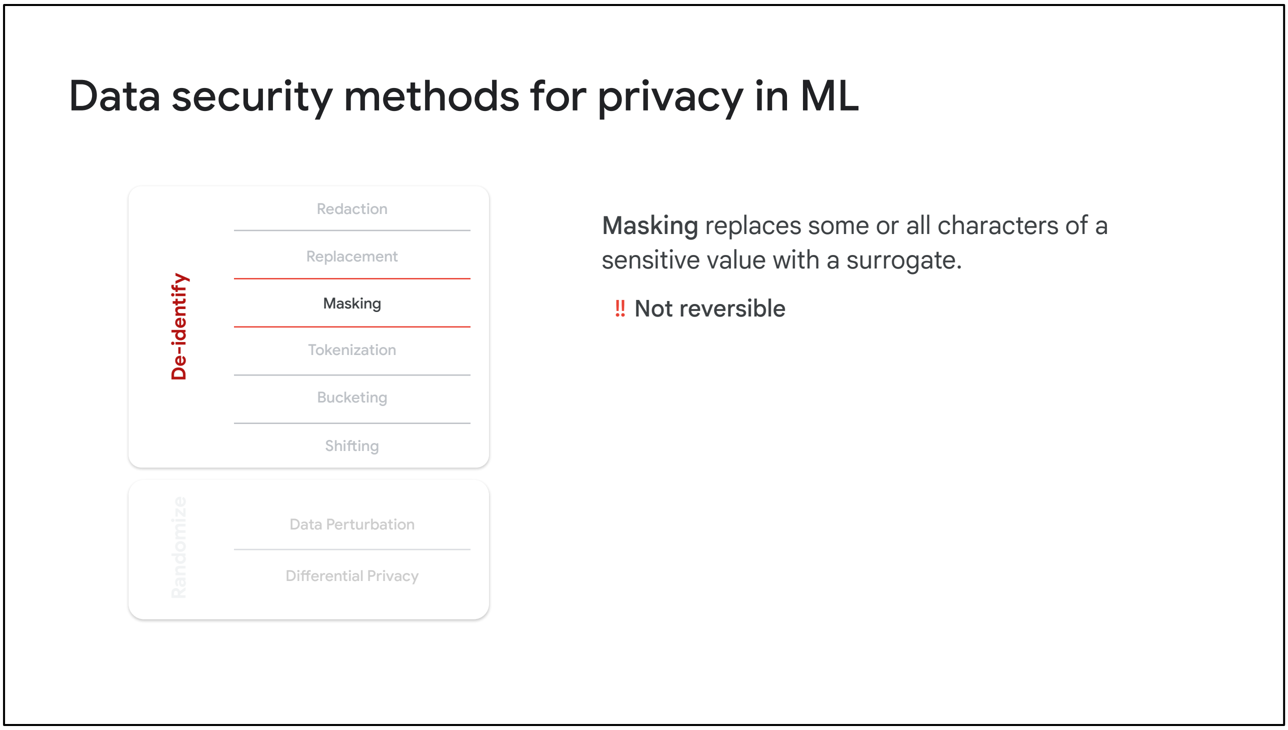
- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Masking replaces some or all characters of a sensitive value with a surrogate.

Masking is the third de-identification technique. Masking replaces some or all characters of a sensitive value with a surrogate.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

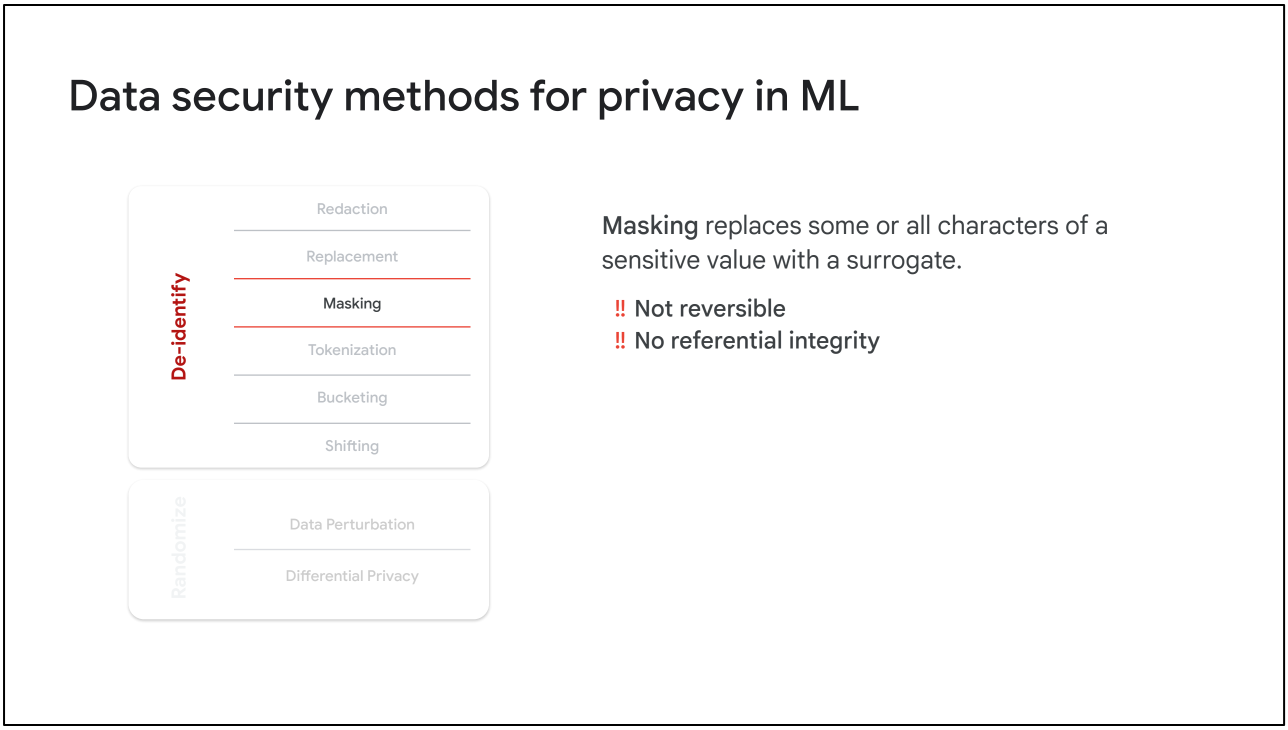
Randomize

- Data Perturbation
- Differential Privacy

Masking replaces some or all characters of a sensitive value with a surrogate.

‼ Not reversible

This surrogate value can be selected randomly from a list if desired, but the approach is not reversible,



Data security methods for privacy in ML

De-identify

Redaction

Replacement

Masking

Tokenization

Bucketing

Shifting

Randomize

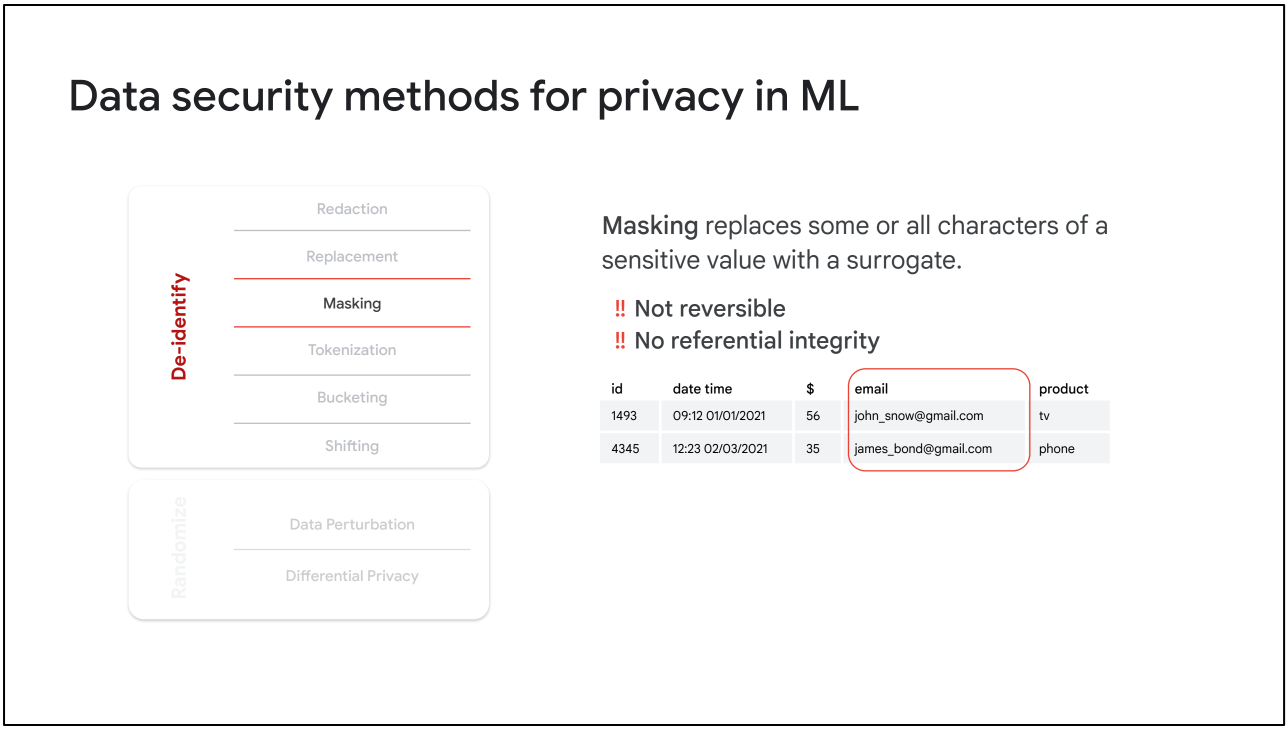
Data Perturbation

Differential Privacy

Masking replaces some or all characters of a sensitive value with a surrogate.

‼ Not reversible
‼ No referential integrity

and it does not maintain referential integrity.



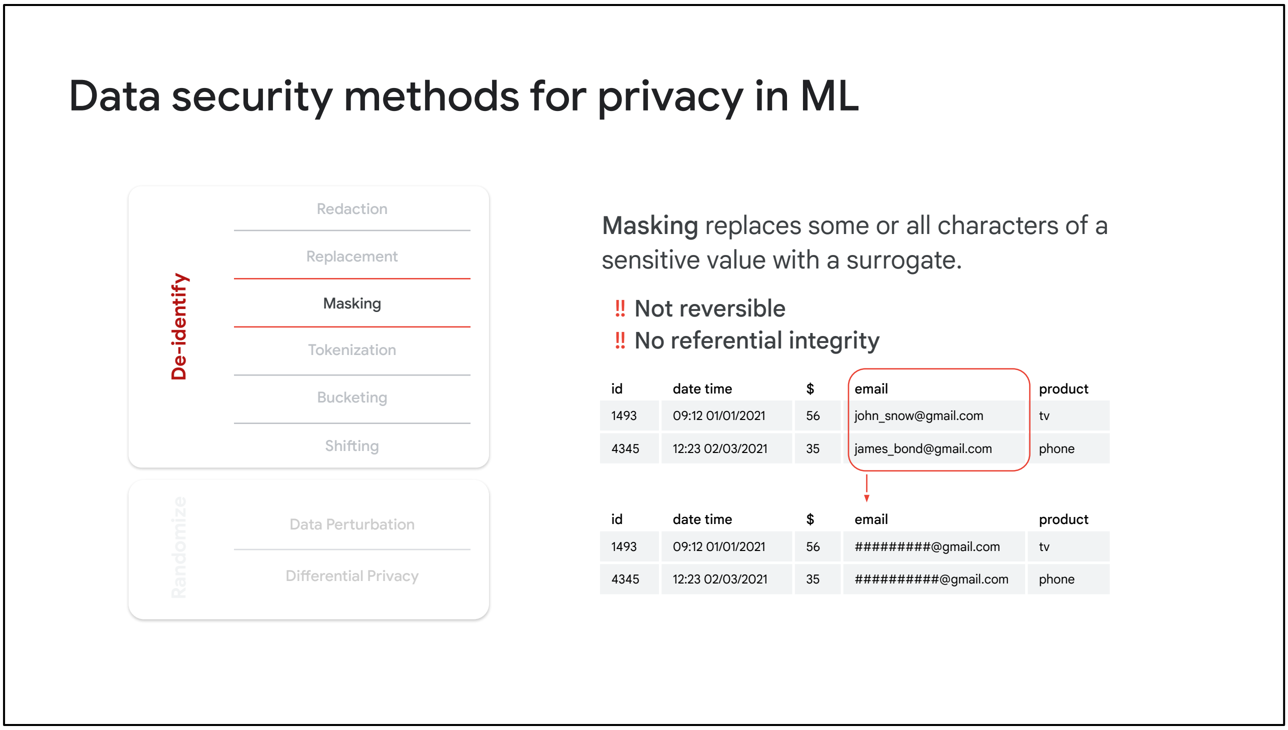
Data security methods for privacy in ML

Masking replaces some or all characters of a sensitive value with a surrogate.

‼ Not reversible
‼ No referential integrity

id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

For example, let’s say you want to apply masking on the email column to keep the email domain but not keep the exact handle,



Data security methods for privacy in ML

De-identify
- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize
- Data Perturbation
- Differential Privacy

Masking replaces some or all characters of a sensitive value with a surrogate.

‼ Not reversible
‼ No referential integrity

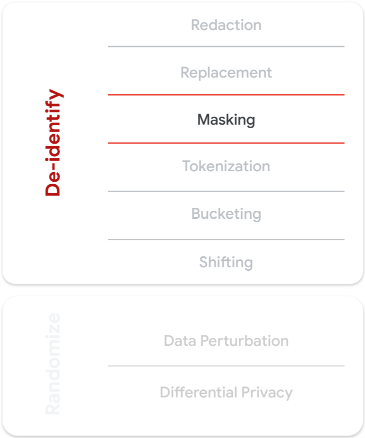
id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

id	date time	$	email	product
1493	09:12 01/01/2021	56	#########@gmail.com	tv
4345	12:23 02/03/2021	35	##########@gmail.com	phone

then you can replace all values in the “email” column with hashes.

While replacement and masking may sound similar, masking covers a portion or all of the existing data, depending on the parameters you set. Replacement removes the data completely and replaces it with a value.

Data security methods for privacy in ML

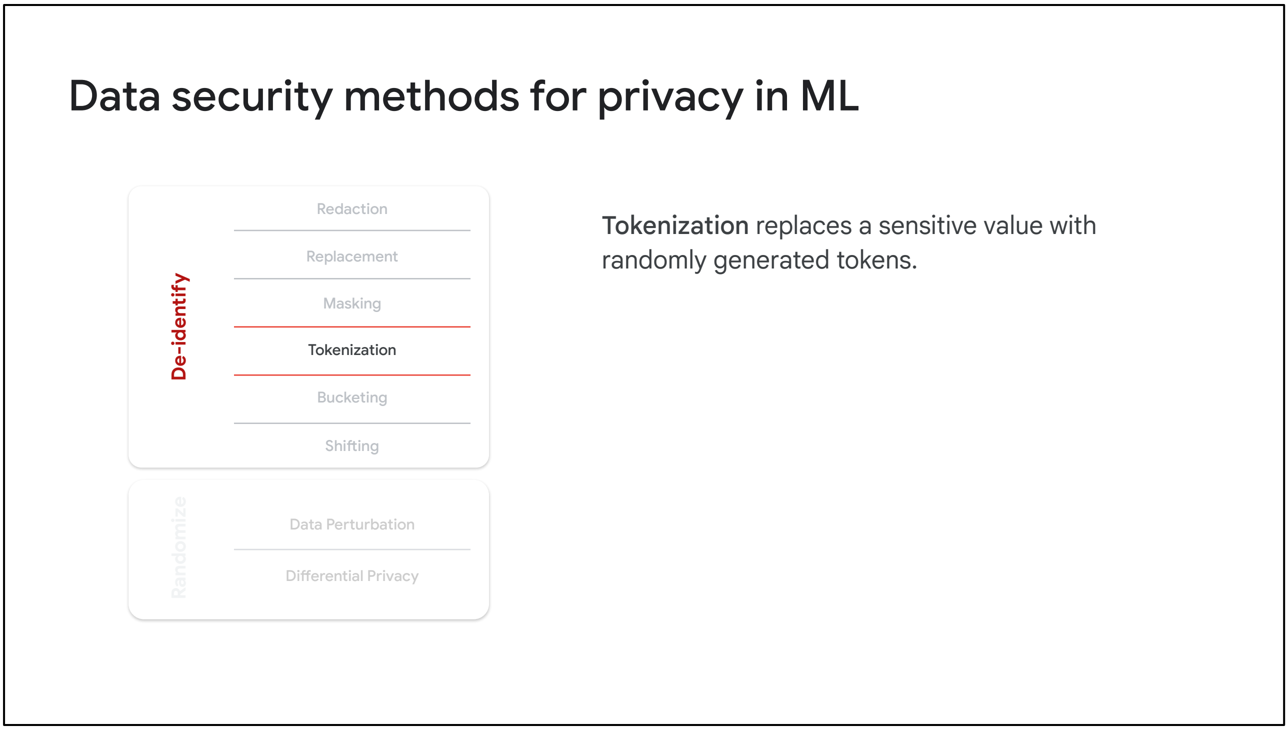


De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy



Data security methods for privacy in ML

De-identify

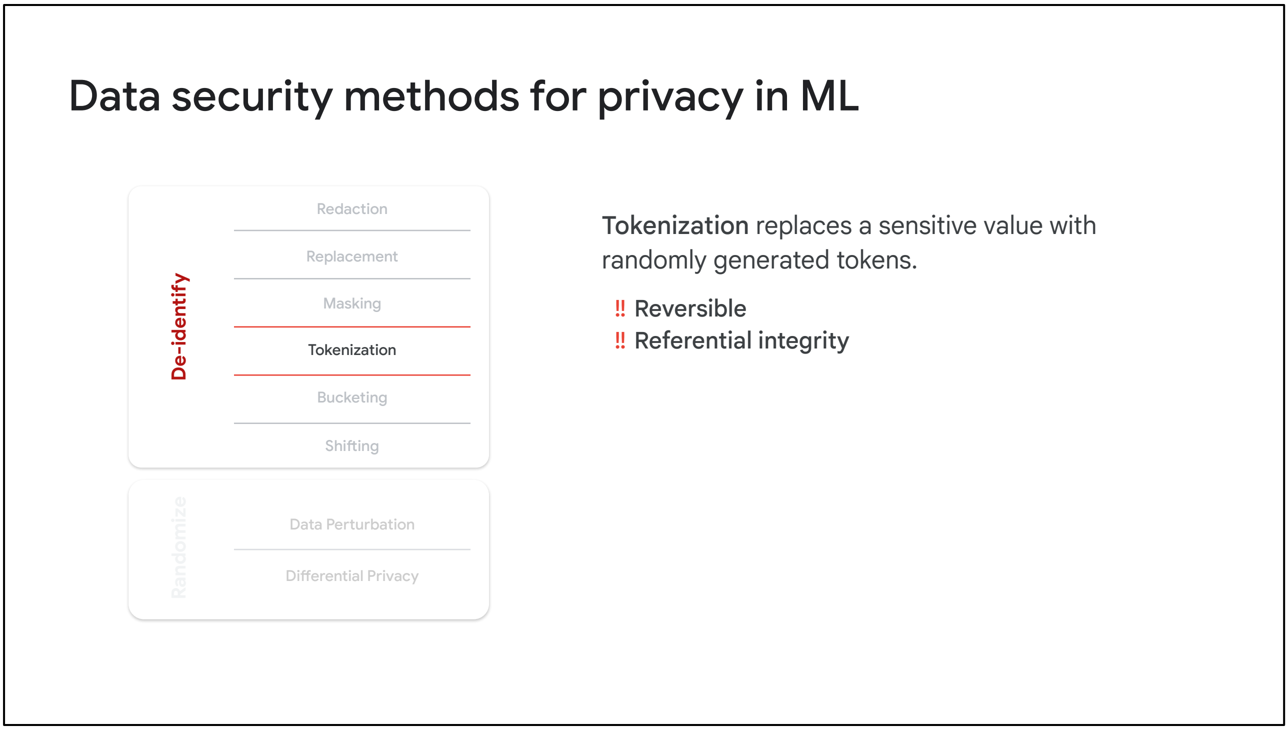
- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Tokenization replaces a sensitive value with randomly generated tokens.

Tokenization is the last de-identification technique we will explore for categoricals. Tokenization replaces a sensitive value with randomly generated tokens – each unique to the value it de-identifies.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

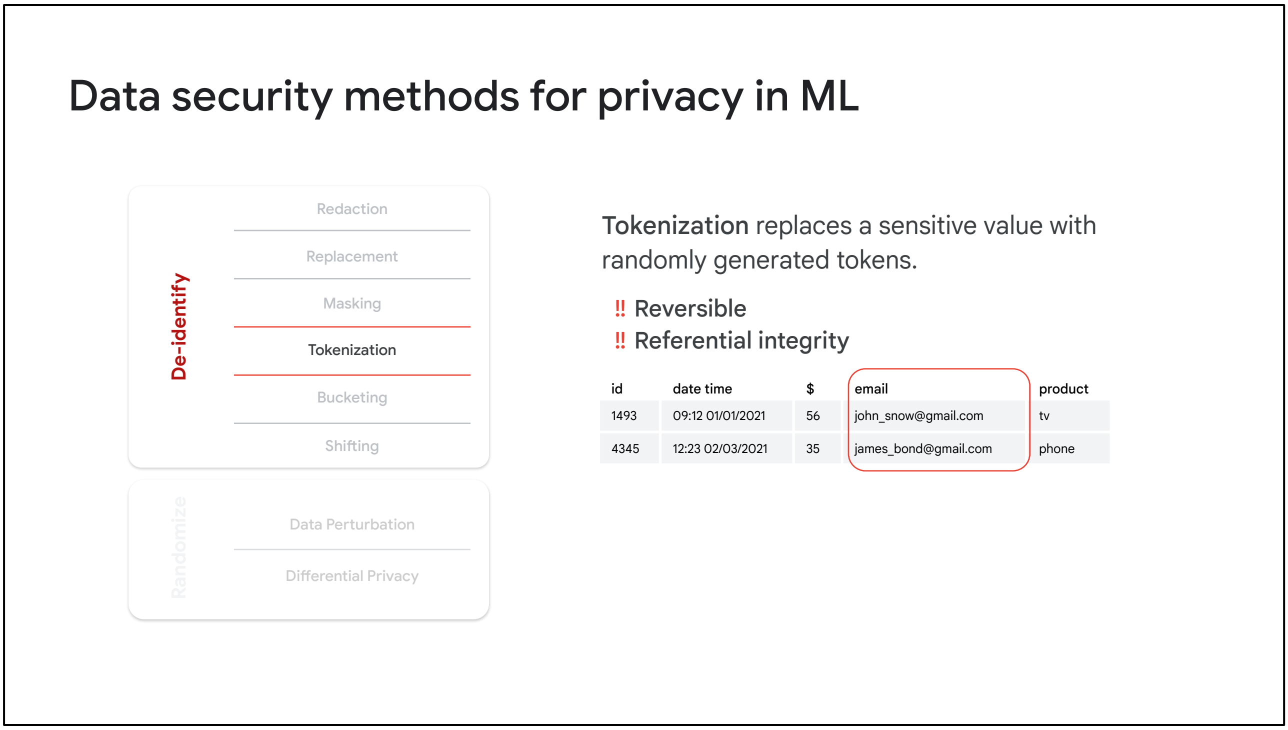
Randomize

- Data Perturbation
- Differential Privacy

Tokenization replaces a sensitive value with randomly generated tokens.

‼ Reversible
‼ Referential integrity

This approach is reversible and also maintains referential integrity where ML models can still leverage information from the tokenized features. However, the data can be vulnerable to malicious attacks that attempt to reverse the data to the original information.



Data security methods for privacy in ML

De-identify

Redaction

Replacement

Masking

Tokenization

Bucketing

Shifting

Randomize

Data Perturbation

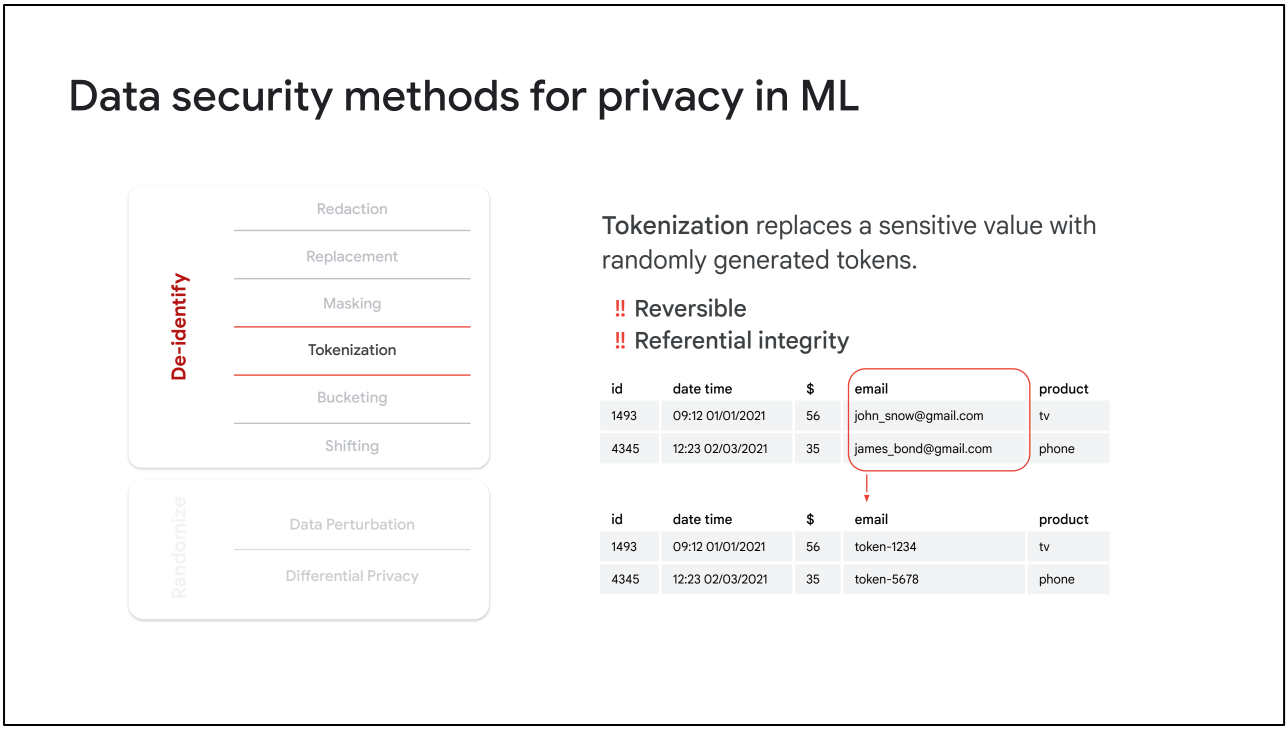
Differential Privacy

Tokenization replaces a sensitive value with randomly generated tokens.

‼ Reversible
‼ Referential integrity

id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

For example, if you apply tokenization on the email column,



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Tokenization replaces a sensitive value with randomly generated tokens.

‼ Reversible
‼ Referential integrity

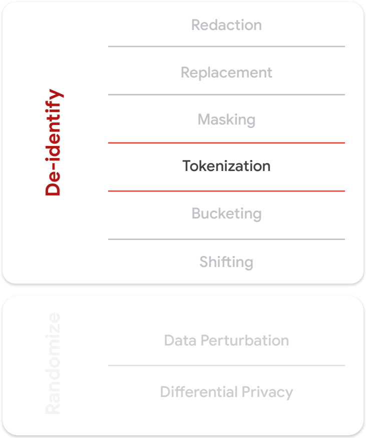
id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

id	date time	$	email	product
1493	09:12 01/01/2021	56	token-1234	tv
4345	12:23 02/03/2021	35	token-5678	phone

then you get new random tokens which uniquely identify that value.

With tokenization, ML models can still leverage the information of tokenized features. But be careful as it can be vulnerable to malicious attacks that reverse it to the original information.

Data security methods for privacy in ML

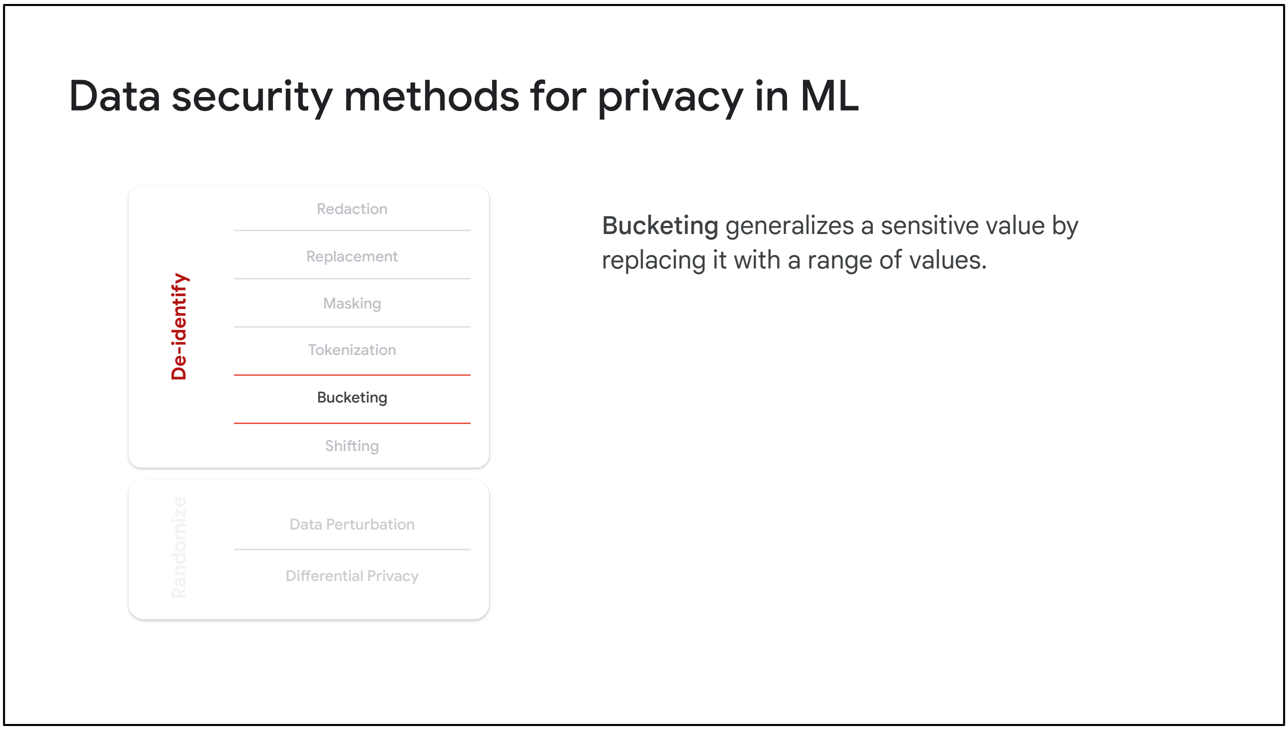


De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy



Data security methods for privacy in ML

De-identify

Redaction

Replacement

Masking

Tokenization

Bucketing

Shifting

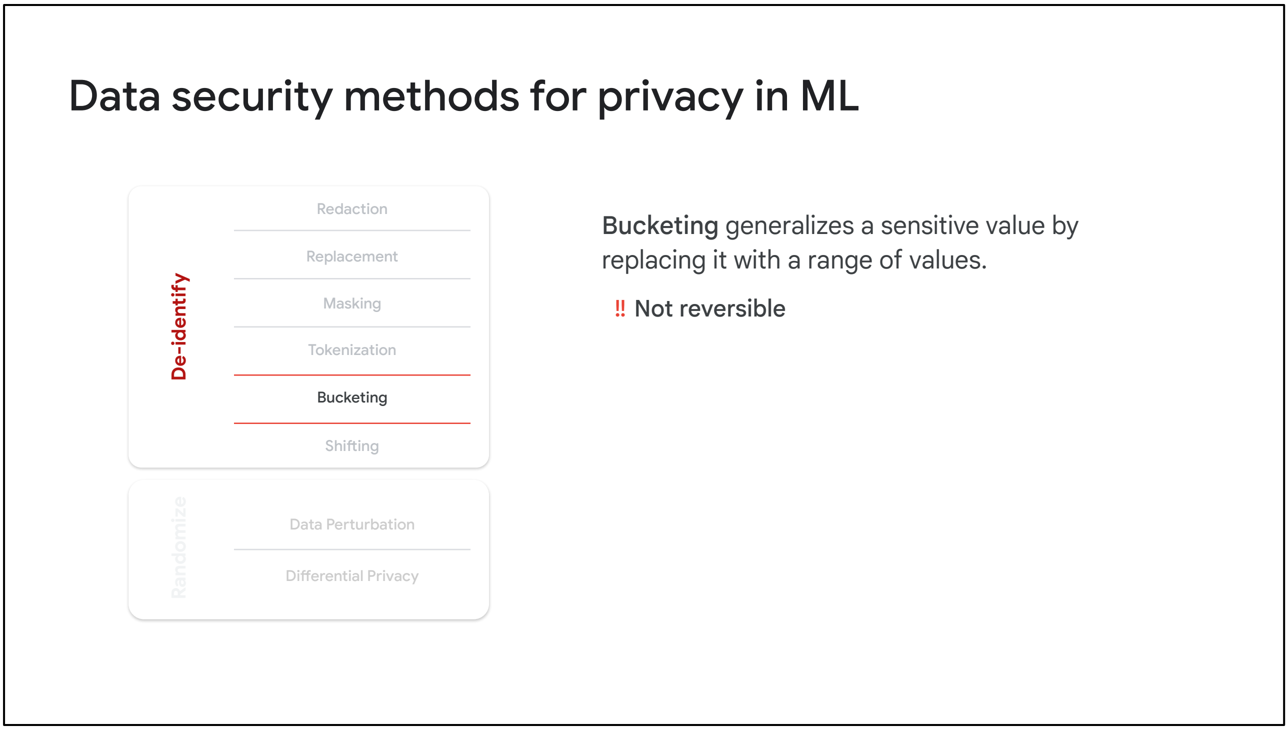
Randomize

Data Perturbation

Differential Privacy

Bucketing generalizes a sensitive value by replacing it with a range of values.

The next de-identification approach, bucketing, is for numerical values. Bucketing generalizes a sensitive value by replacing it with a range of values.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- **Bucketing**
- Shifting

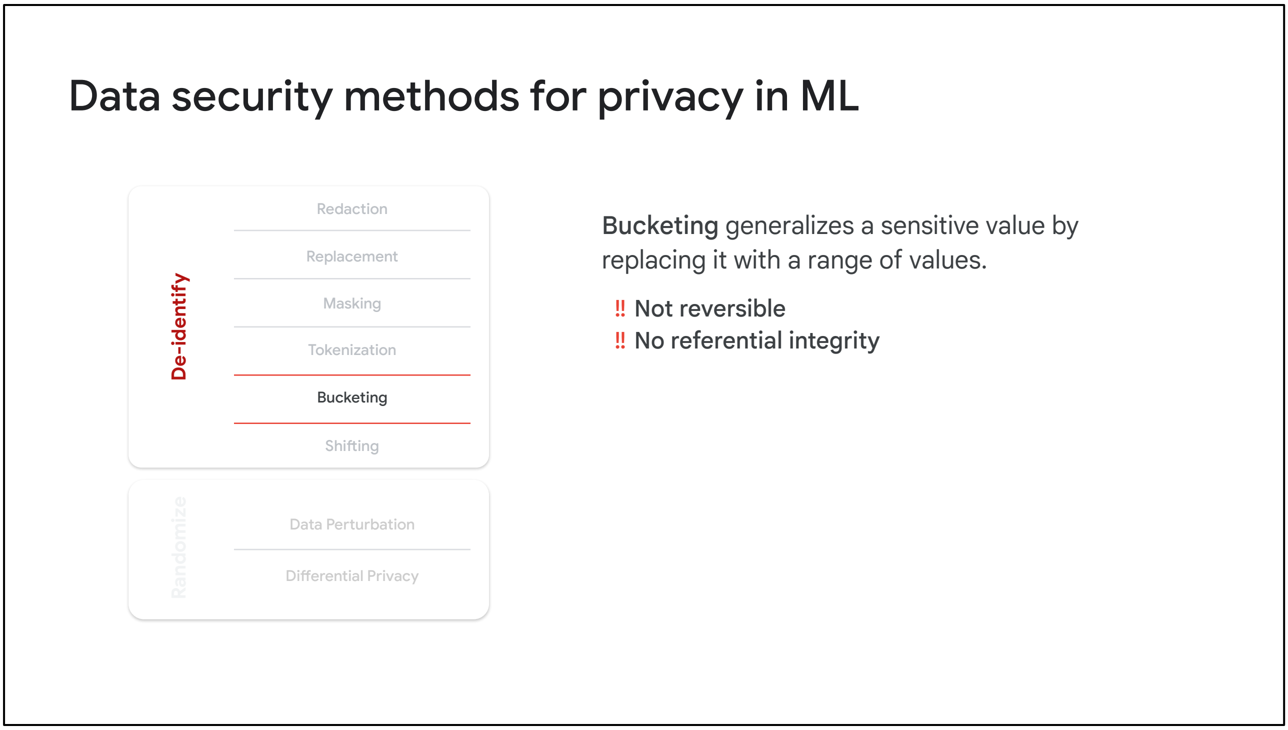
Randomize

- Data Perturbation
- Differential Privacy

Bucketing generalizes a sensitive value by replacing it with a range of values.

‼ Not reversible

This approach is not reversible,



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- **Bucketing**
- Shifting

Randomize

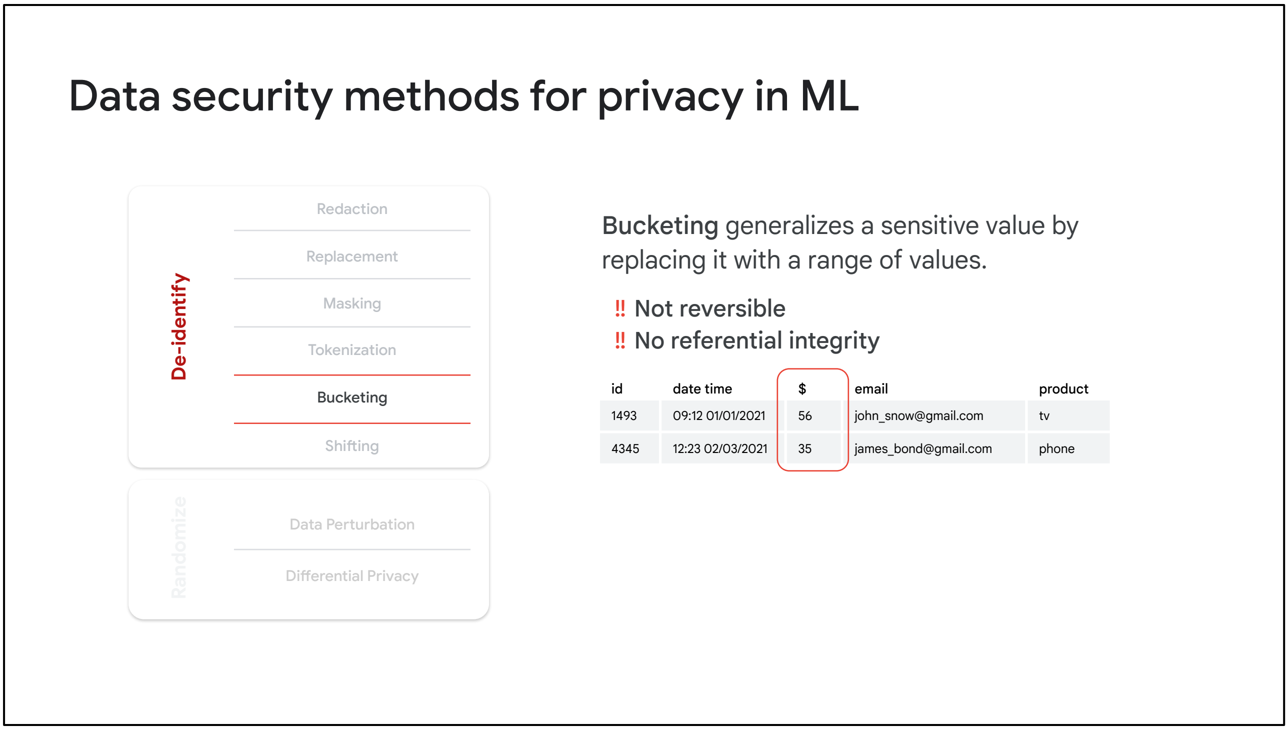
- Data Perturbation
- Differential Privacy

Bucketing generalizes a sensitive value by replacing it with a range of values.

‼ Not reversible

‼ No referential integrity

and it does not maintain referential integrity.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- **Bucketing**
- Shifting

Randomize

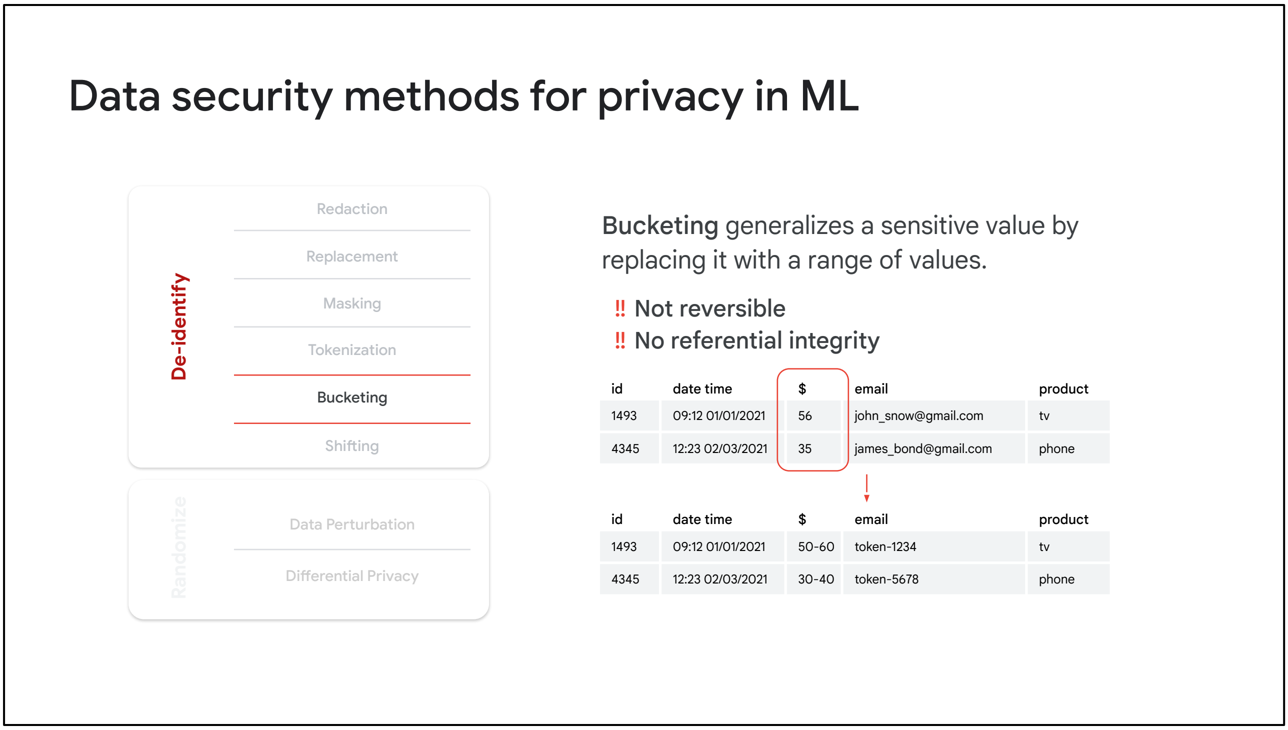
- Data Perturbation
- Differential Privacy

Bucketing generalizes a sensitive value by replacing it with a range of values.

‼ Not reversible
‼ No referential integrity

id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

For example, let’s say you want to apply bucketing to the amount in dollars column,



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Bucketing generalizes a sensitive value by replacing it with a range of values.

‼ Not reversible
‼ No referential integrity

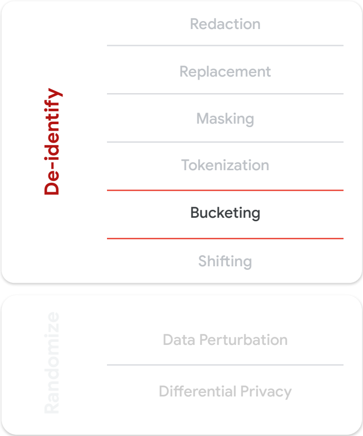
id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

id	date time	$	email	product
1493	09:12 01/01/2021	50-60	token-1234	tv
4345	12:23 02/03/2021	30-40	token-5678	phone

and you want to bucket in tens. Then the procedure would replace each entry with its correct bucket, e.g., 56 is replaced with 50-60.

Depending on how broadly the buckets are defined, the de-identified data may be less useful for machine learning purposes or may affect the model performance. Be careful about the size and granularity of brackets so that you can find a good balance between the model performance and privacy.

Data security methods for privacy in ML

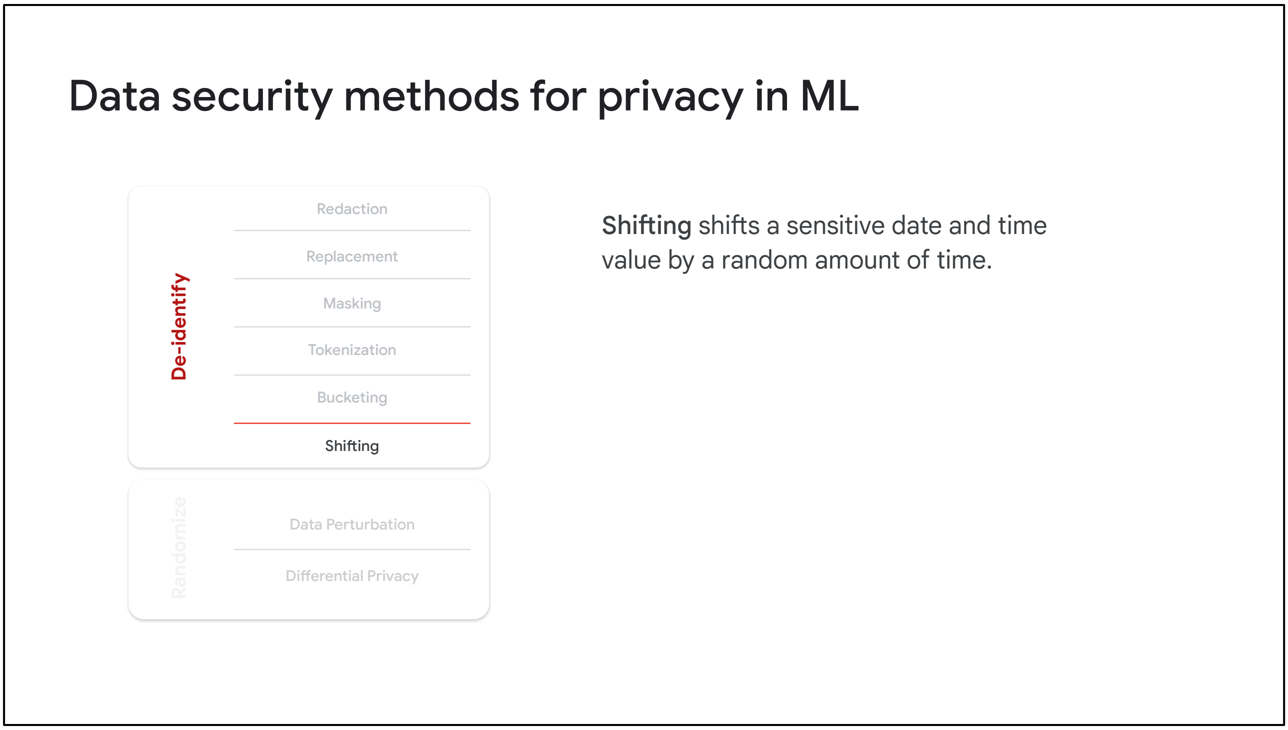


De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy



Data security methods for privacy in ML

De-identify

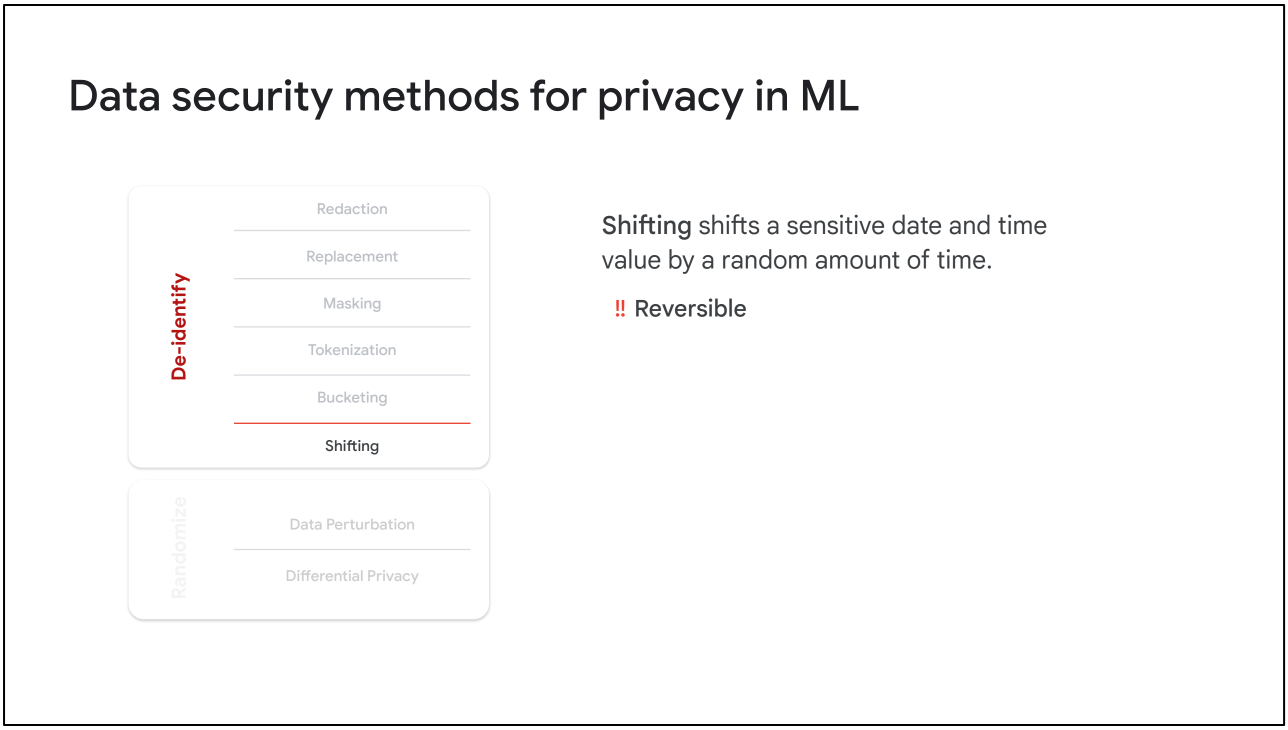
- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Shifting shifts a sensitive date and time value by a random amount of time.

Lastly, shifting, which is a technique for datetime values.
This method shifts a sensitive date and time value by a random amount of time.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- **Shifting**

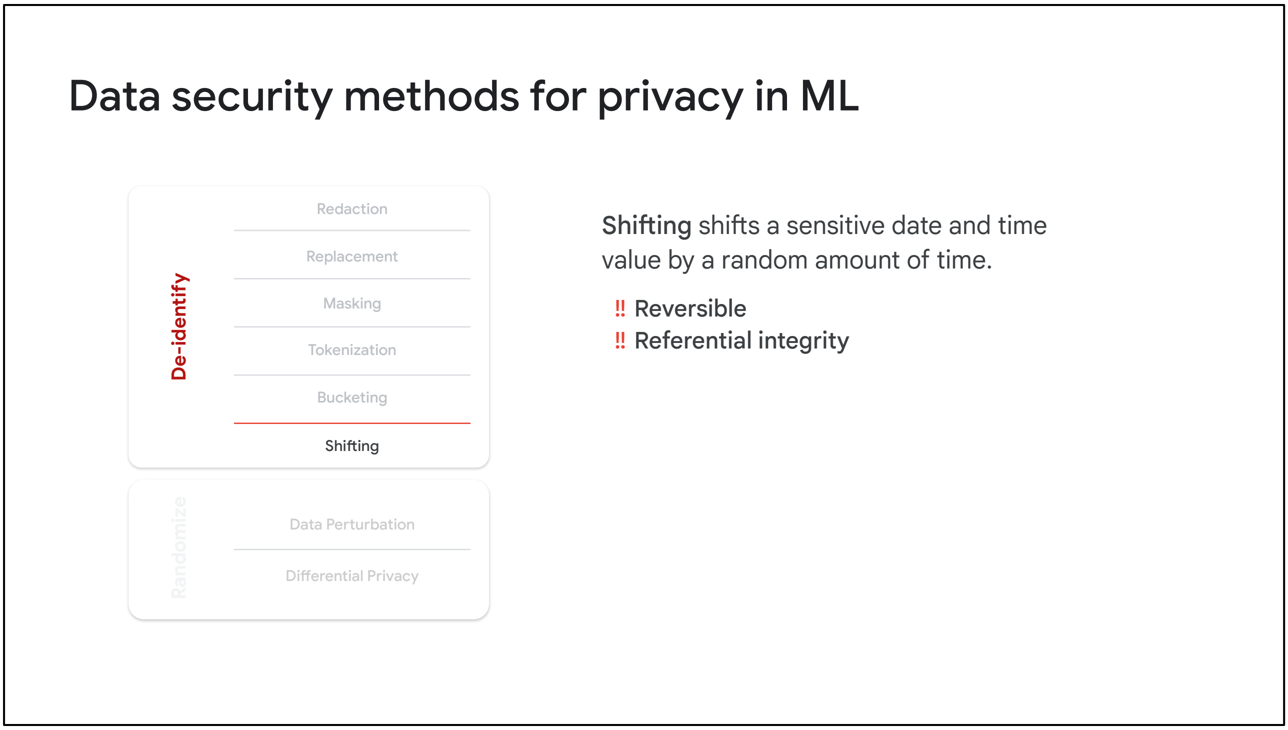
Randomize

- Data Perturbation
- Differential Privacy

Shifting shifts a sensitive date and time value by a random amount of time.

‼ Reversible

This transformation is reversible,



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

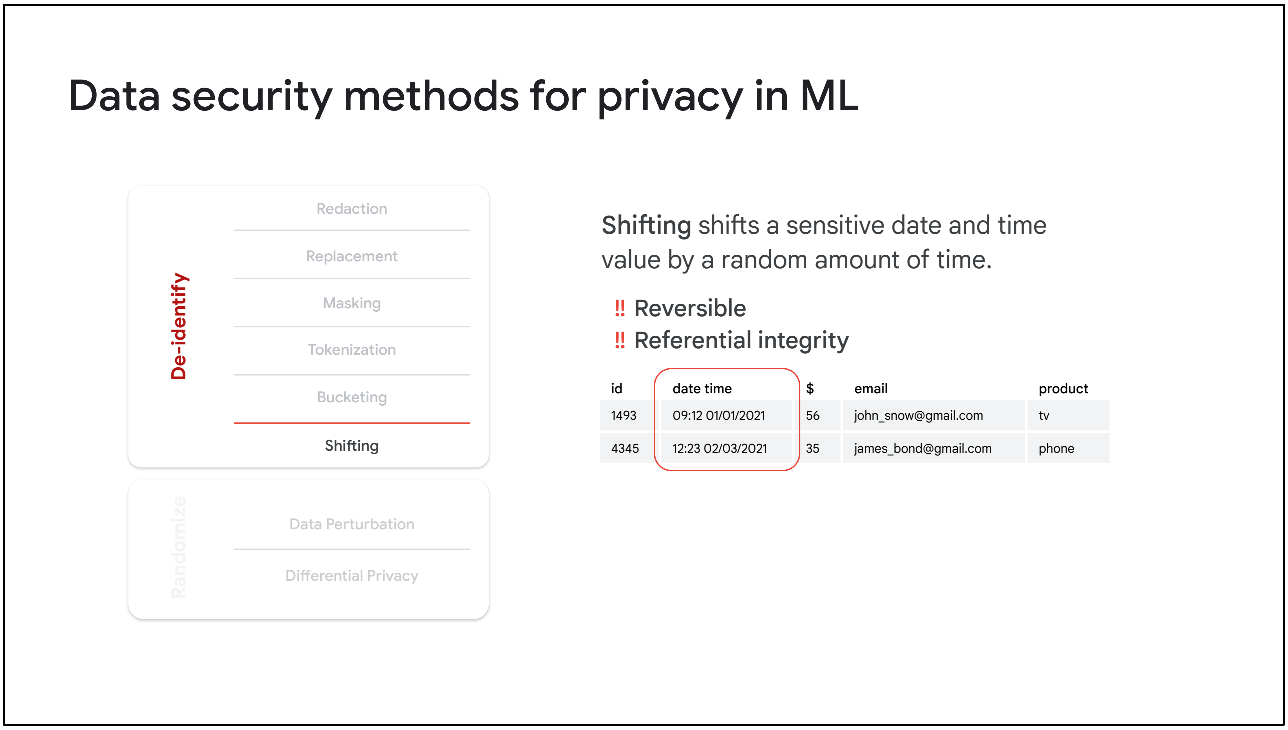
- Data Perturbation
- Differential Privacy

Shifting shifts a sensitive date and time value by a random amount of time.

‼ Reversible

‼ Referential integrity

and it keeps referential integrity because sequence and duration are kept.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- **Shifting**

Randomize

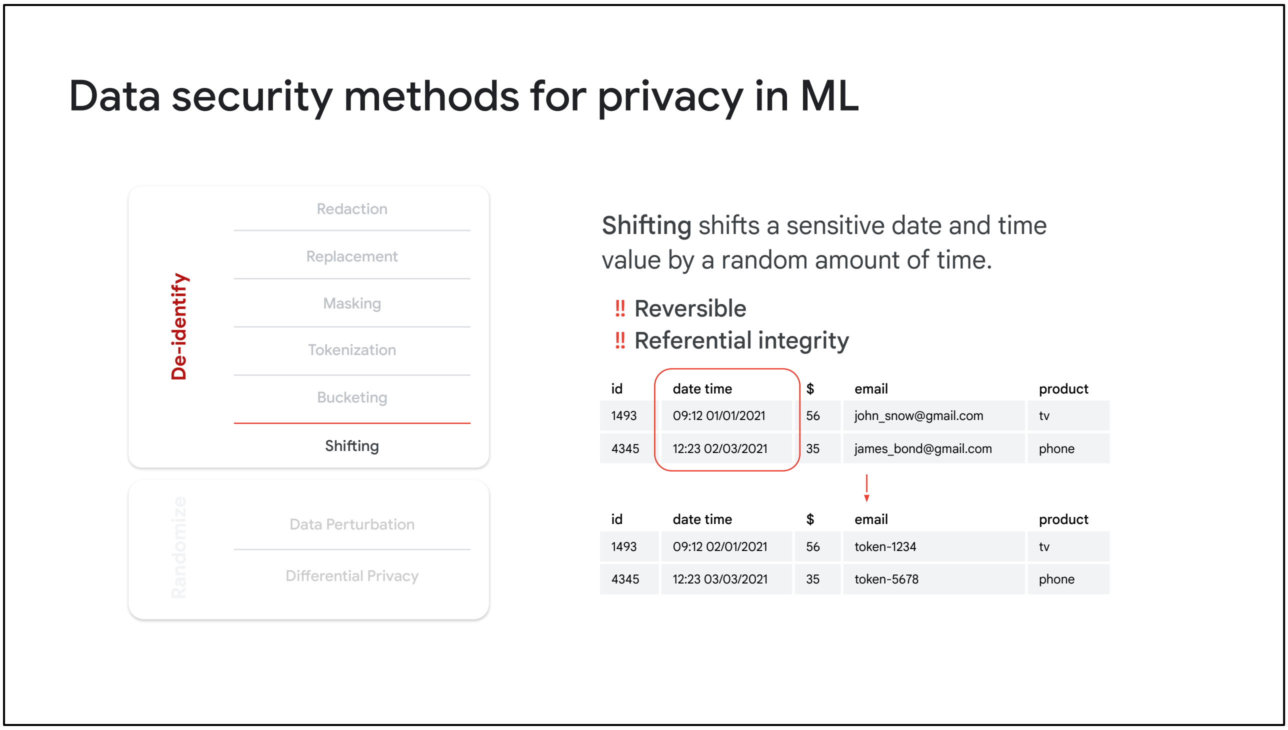
- Data Perturbation
- Differential Privacy

Shifting shifts a sensitive date and time value by a random amount of time.

‼ Reversible
‼ Referential integrity

id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

For example, if you apply shifting to the datetime column,



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- **Shifting**

Randomize

- Data Perturbation
- Differential Privacy

Shifting shifts a sensitive date and time value by a random amount of time.

‼ Reversible
‼ Referential integrity

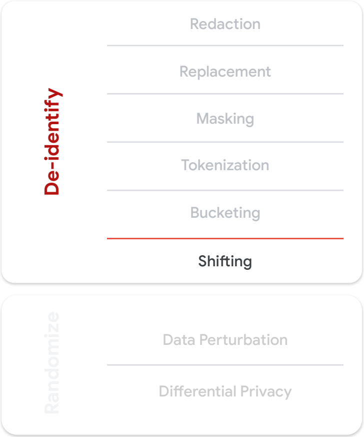
id	date time	$	email	product
1493	09:12 01/01/2021	56	john_snow@gmail.com	tv
4345	12:23 02/03/2021	35	james_bond@gmail.com	phone

id	date time	$	email	product
1493	09:12 02/01/2021	56	token-1234	tv
4345	12:23 03/03/2021	35	token-5678	phone

then the procedure could randomly select one day as shift, resulting in one day added to each entry.

Similar to tokenization, shifting can also be vulnerable to malicious attacks if the random shifting factor is identified.

Data security methods for privacy in ML

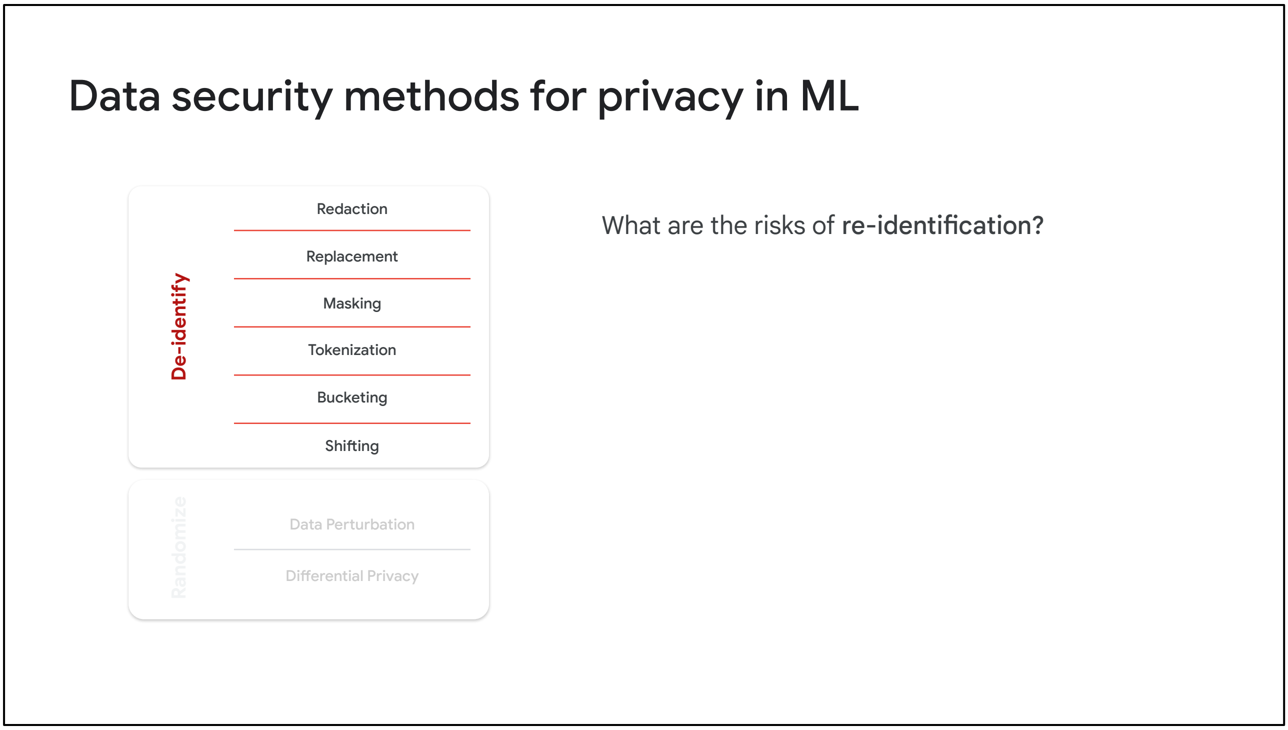


De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

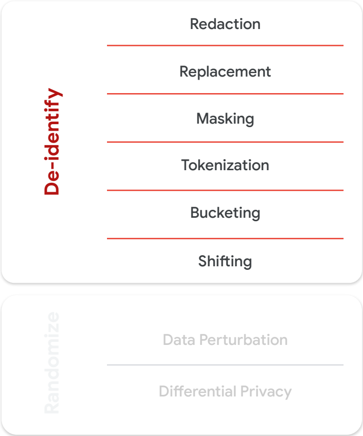
- Data Perturbation
- Differential Privacy

What are the risks of **re-identification?**

Now, what are the risks of re-identification?

None of these approaches guarantee 100% that data cannot be re-identified — even the non-reversible approaches. Instead, what these approaches do is strive towards minimizing the risk.

Data security methods for privacy in ML

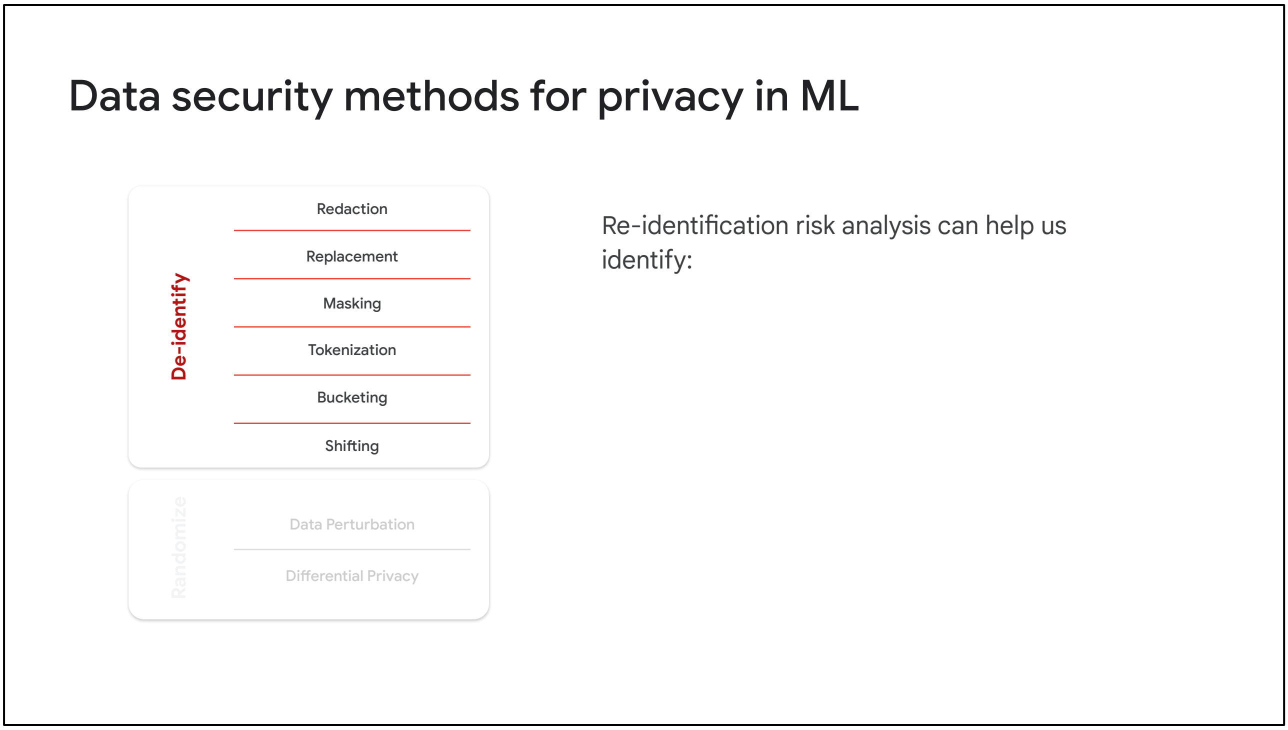


De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy



Data security methods for privacy in ML

De-identify

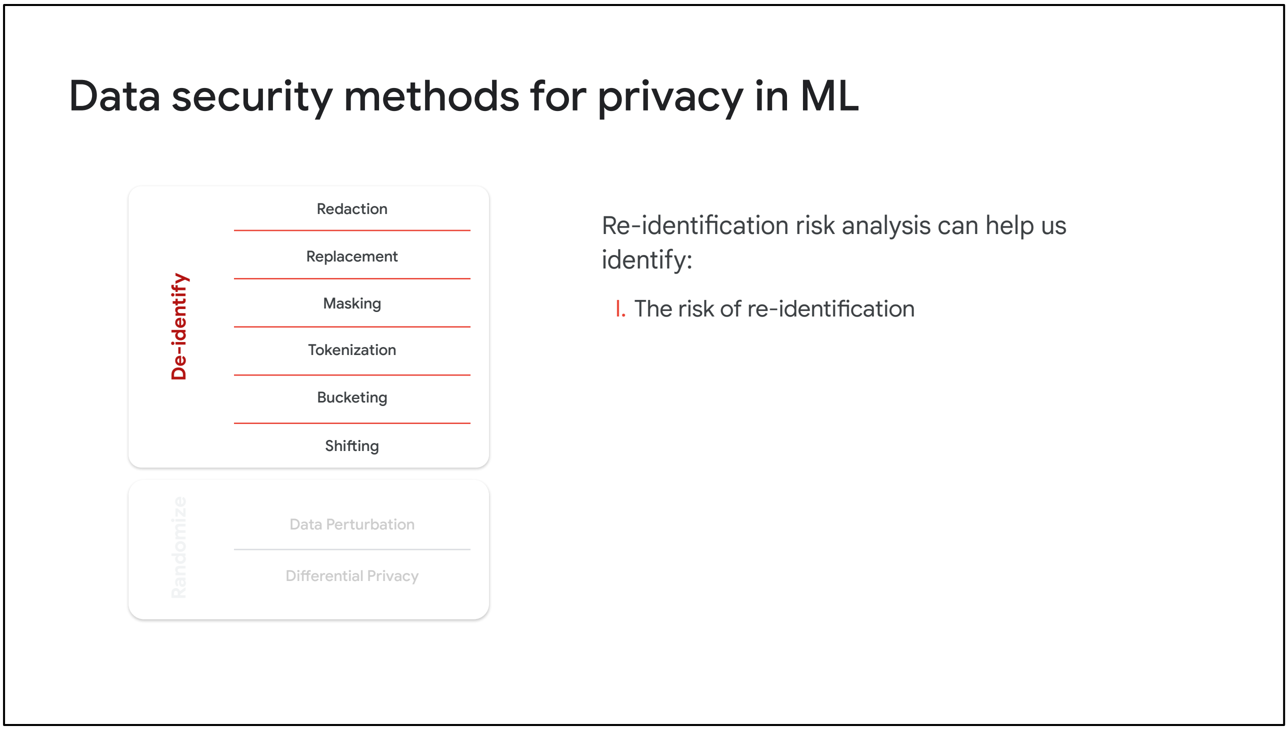
- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Re-identification risk analysis can help us identify:

Re-identification risk analysis can help us:



Data security methods for privacy in ML

De-identify

Redaction

Replacement

Masking

Tokenization

Bucketing

Shifting

Randomize

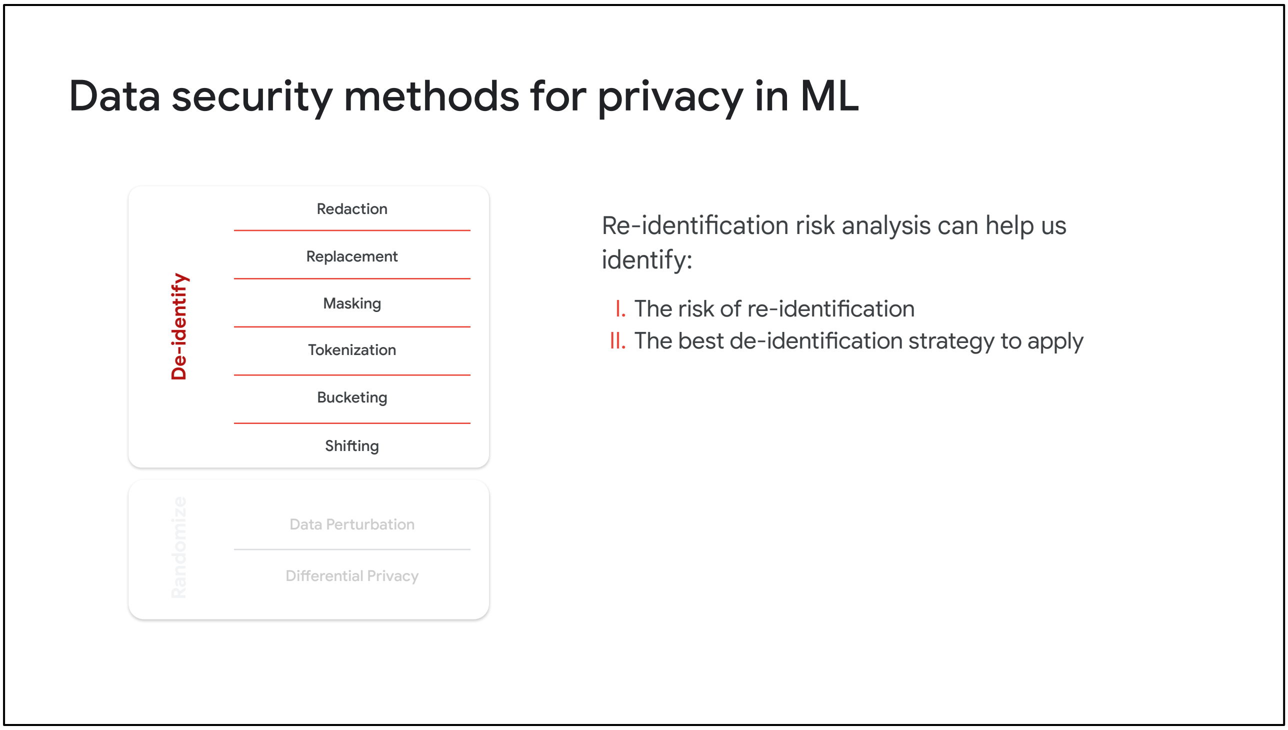
Data Perturbation

Differential Privacy

Re-identification risk analysis can help us identify:

I. The risk of re-identification

- Identify the risk of re-identification after performing a de-identification technique.



Data security methods for privacy in ML

De-identify

Redaction

Replacement

Masking

Tokenization

Bucketing

Shifting

Randomize

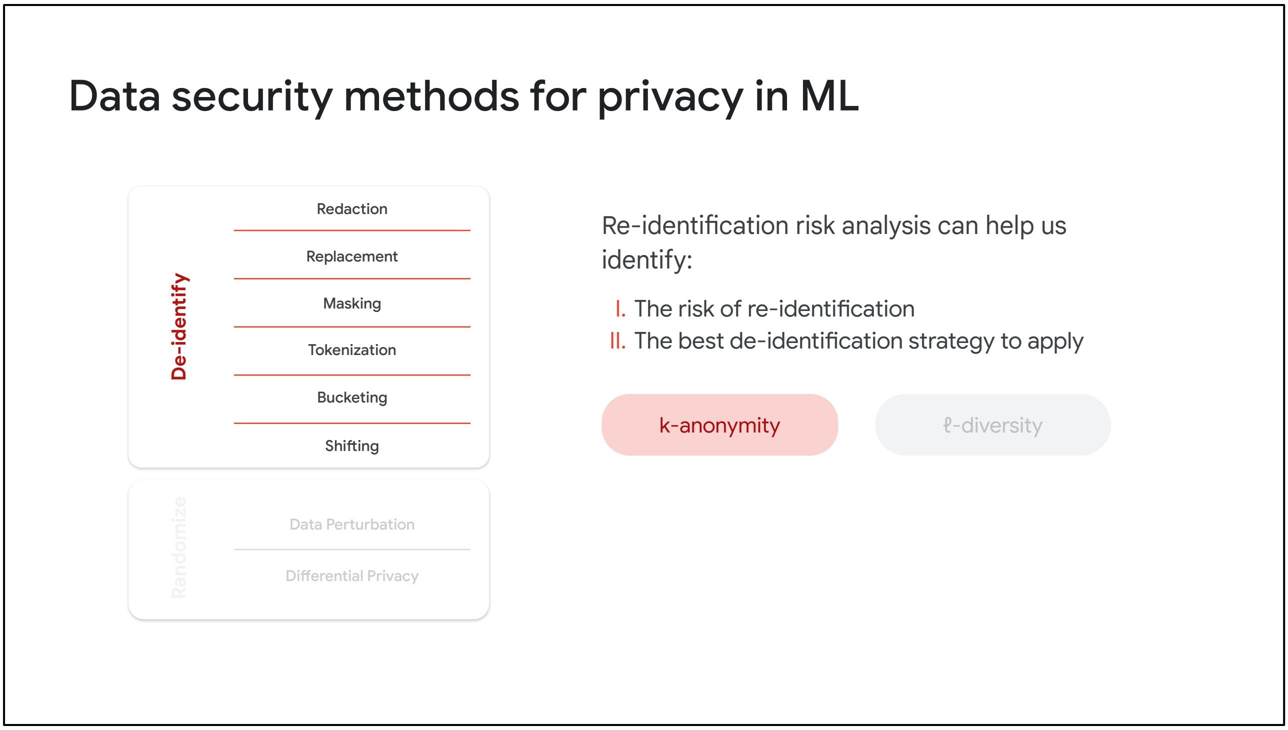
Data Perturbation

Differential Privacy

Re-identification risk analysis can help us identify:

I. The risk of re-identification
II. The best de-identification strategy to apply

- Identify the most effective de-identification strategy before de-identification



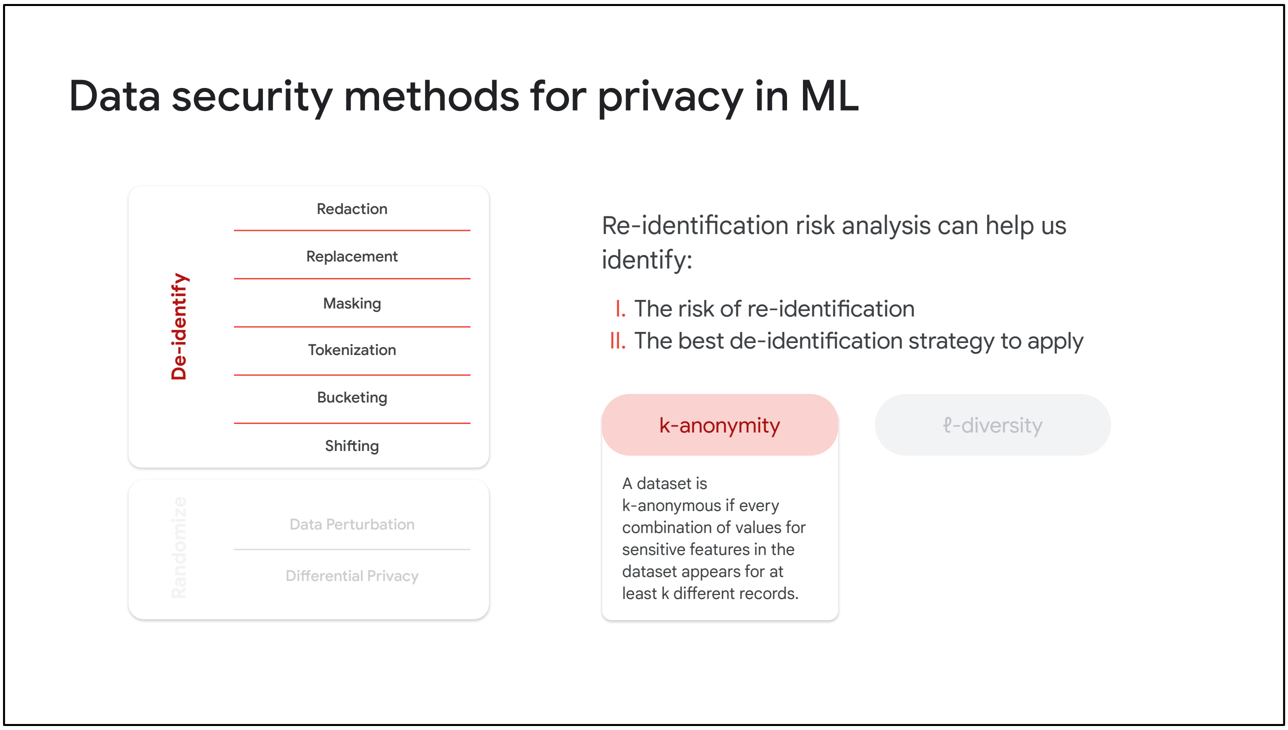
Data security methods for privacy in ML

Re-identification risk analysis can help us identify:

I. The risk of re-identification
II. The best de-identification strategy to apply

There are 2 classic techniques that support this risk analysis:

- K-anonymity.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Re-identification risk analysis can help us identify:

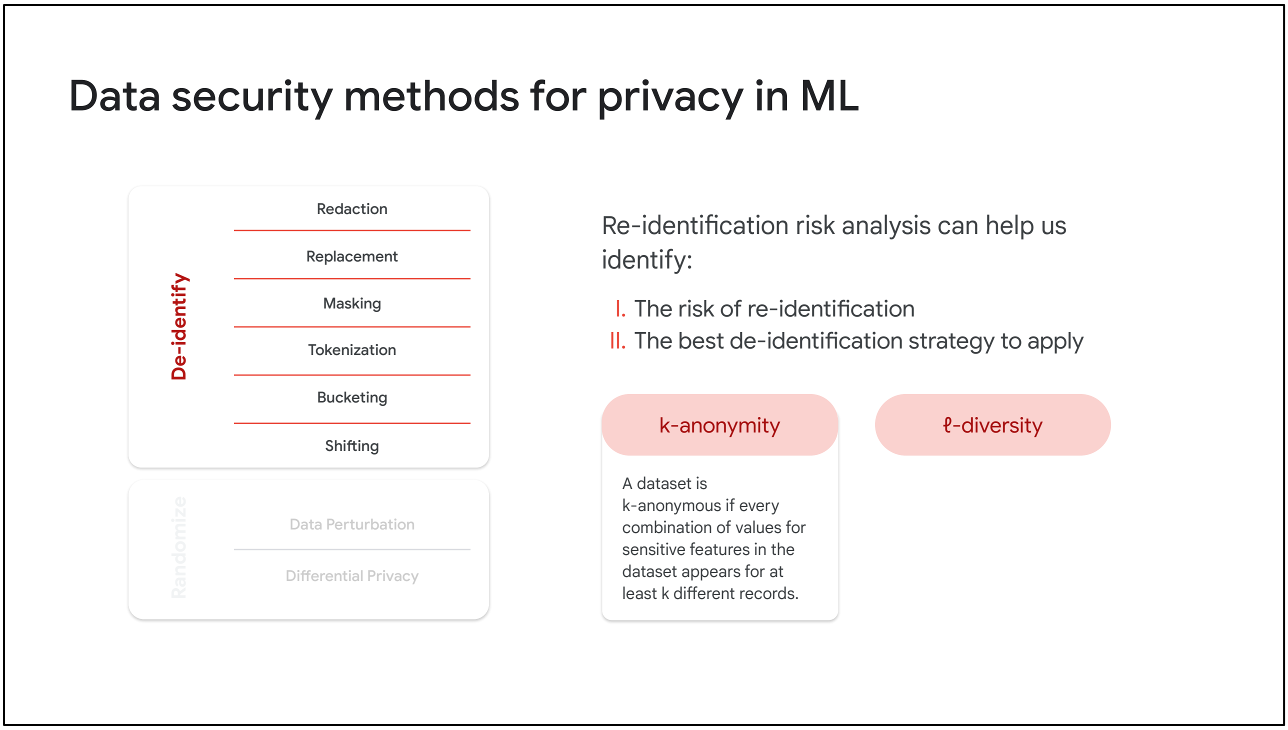
I. The risk of re-identification
II. The best de-identification strategy to apply

k-anonymity

ℓ-diversity

A dataset is k-anonymous if every combination of values for sensitive features in the dataset appears for at least k different records.

A dataset is k-anonymous if every combination of values for sensitive features in the dataset appears for at least k different records.
This tells us something important: features can be correlated, and even if you de-identify a sensitive value, it can still be re-identified given its relationship to other features.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Re-identification risk analysis can help us identify:

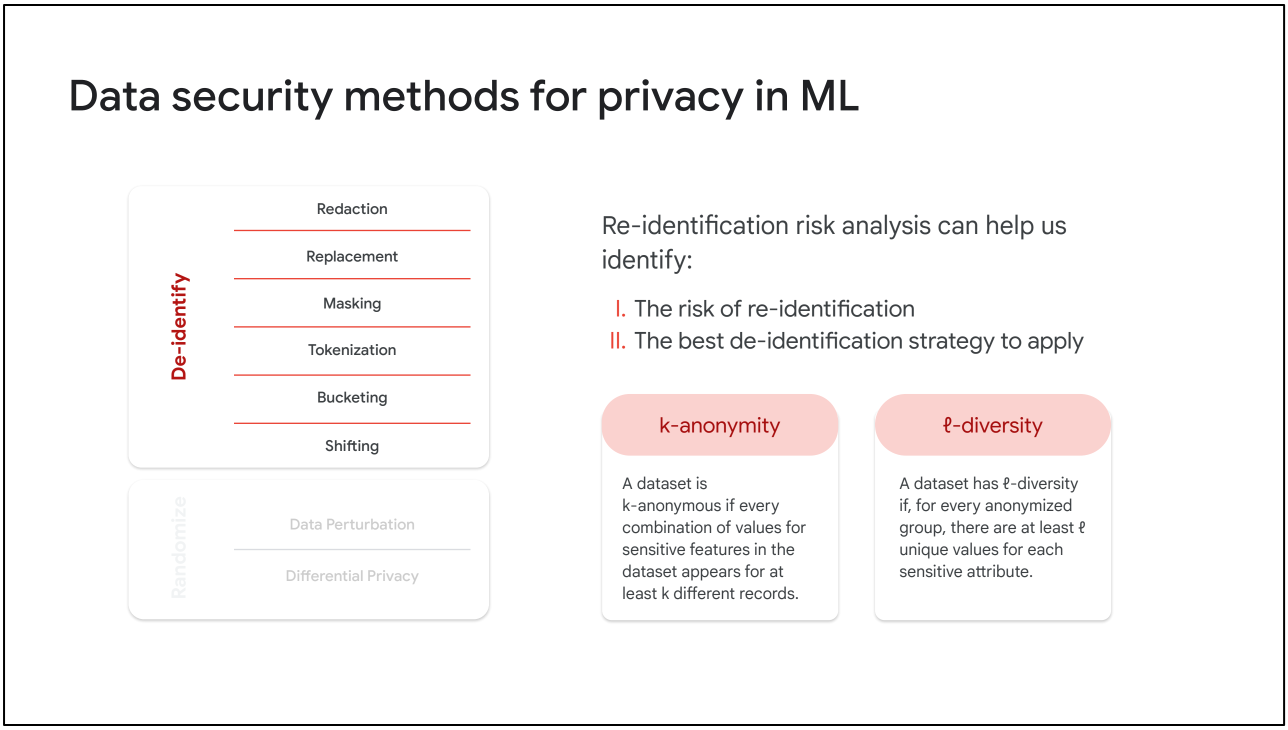
I. The risk of re-identification
II. The best de-identification strategy to apply

k-anonymity

A dataset is k-anonymous if every combination of values for sensitive features in the dataset appears for at least k different records.

ℓ-diversity

- L-diversity.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Re-identification risk analysis can help us identify:

I. The risk of re-identification
II. The best de-identification strategy to apply

k-anonymity

A dataset is k-anonymous if every combination of values for sensitive features in the dataset appears for at least k different records.

ℓ-diversity

A dataset has ℓ-diversity if, for every anonymized group, there are at least ℓ unique values for each sensitive attribute.

Is an extension of k-anonymity that ensures that each anonymized group contains at least L distinctive values for each sensitive attribute.
L-diversity is a measurement of the diversity of sensitive data.

You can apply de-identification, as well as data perturbation techniques, to achieve k-anonymity at a specific k and/or l-diversity at a specific l. The combination of k-anonymity and l-diversity provides a more comprehensive understanding of privacy, especially in scenarios where datasets contain mixed data types.

01 Overview of AI Privacy

02 Privacy in Training Data: De-identification techniques

03 Privacy in Training Data: Randomization techniques

04 Privacy in Machine Learning Training: DP-SGD

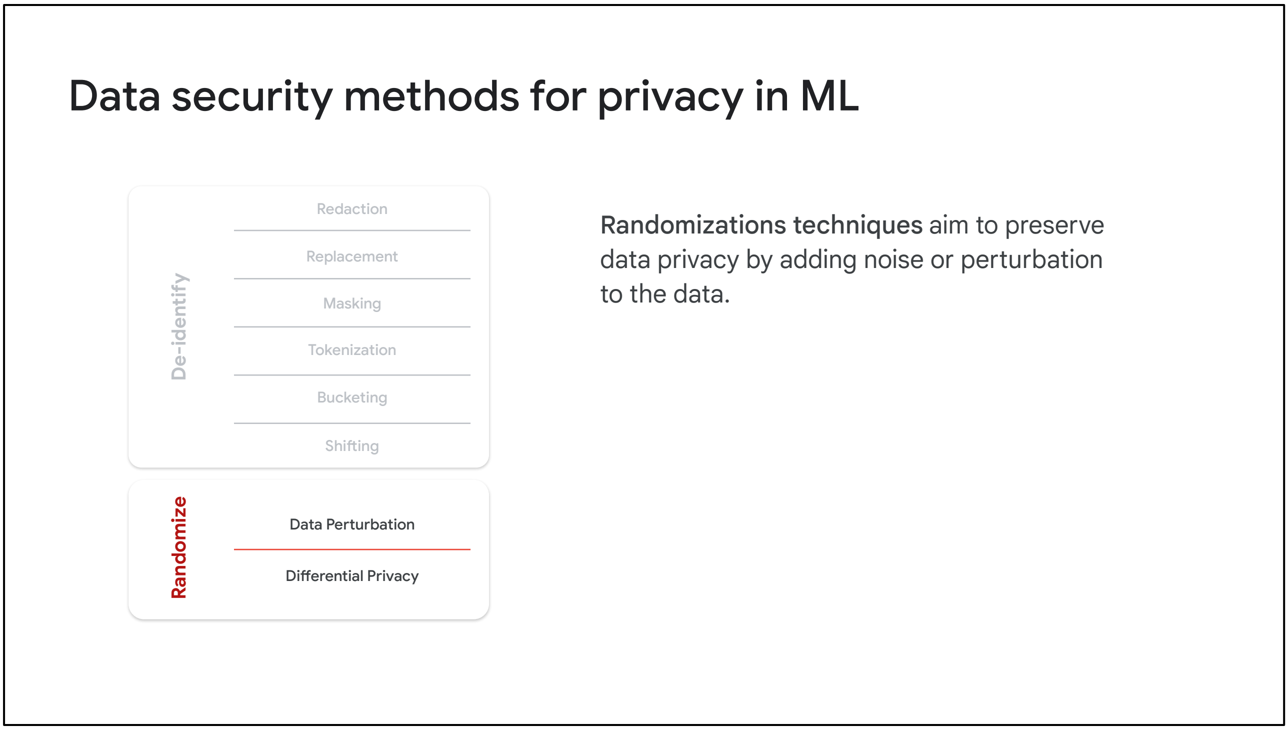
05 Privacy in Machine Learning Training: Federated Learning

06 System Security on Google Cloud

07 System Security and Generative AI

08 Lab: Differential Privacy in Machine Learning with TensorFlow Privacy

Let’s move to randomization techniques for privacy training in data.



Data security methods for privacy in ML

De-identify

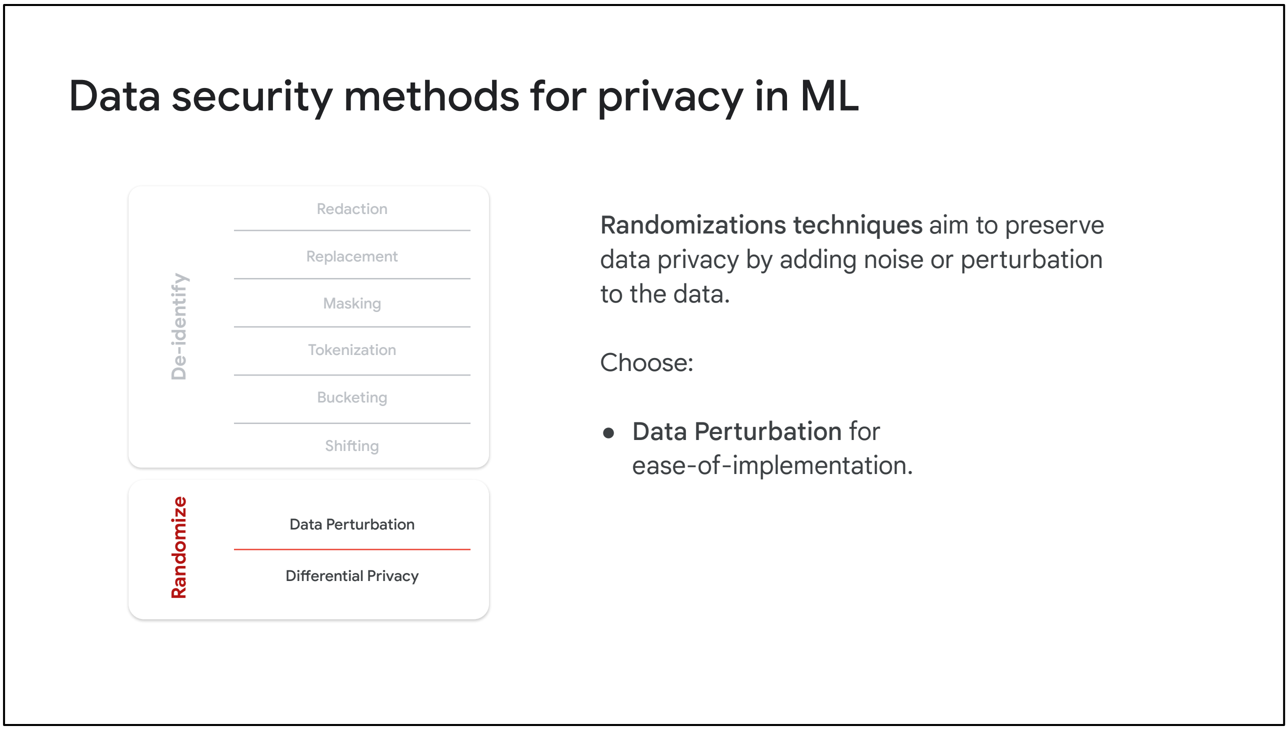
- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Randomizations techniques aim to preserve data privacy by adding noise or perturbation to the data.

Randomization techniques aim to preserve data privacy by adding noise or perturbation to the data.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

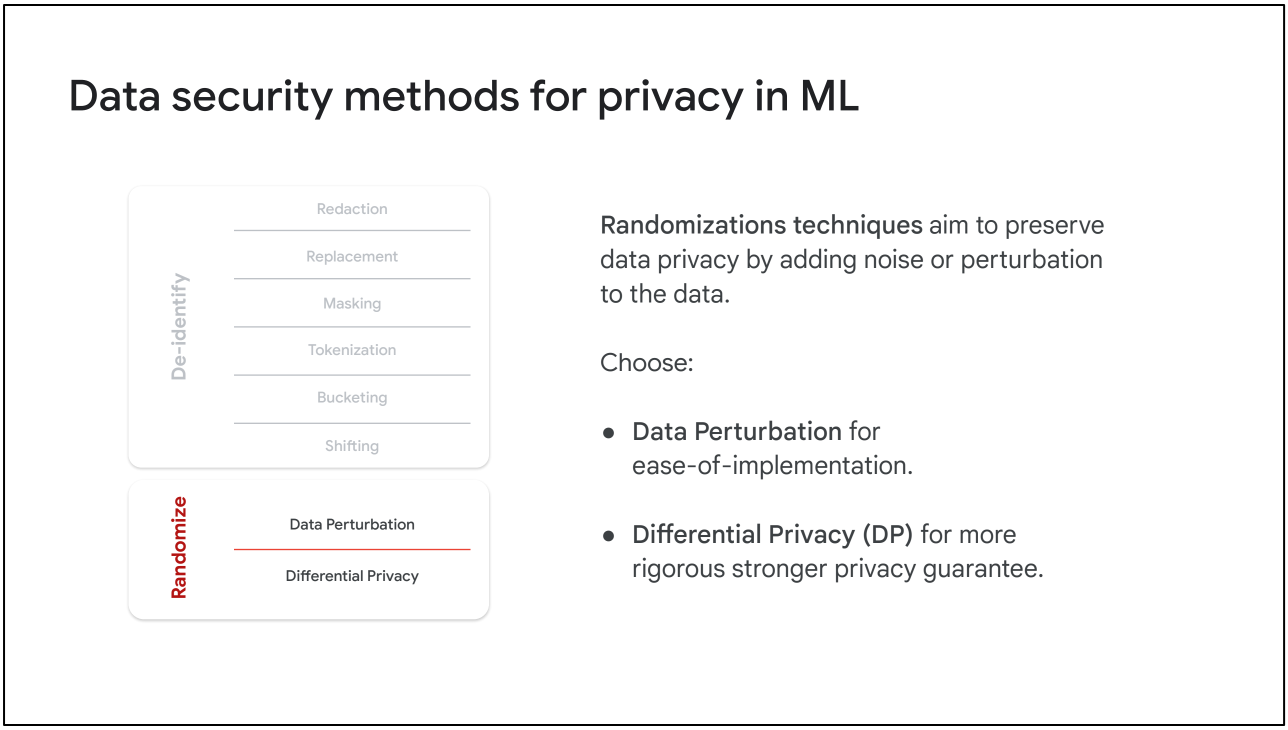
- Data Perturbation
- Differential Privacy

Randomizations techniques aim to preserve data privacy by adding noise or perturbation to the data.

Choose:

- **Data Perturbation** for ease-of-implementation.

When preserving data privacy, you can apply techniques like data perturbation or differential privacy. Data perturbation is a technique you can implement easily.



Data security methods for privacy in ML

De-identify

Redaction

Replacement

Masking

Tokenization

Bucketing

Shifting

Randomize

Data Perturbation

Differential Privacy

Randomizations techniques aim to preserve data privacy by adding noise or perturbation to the data.

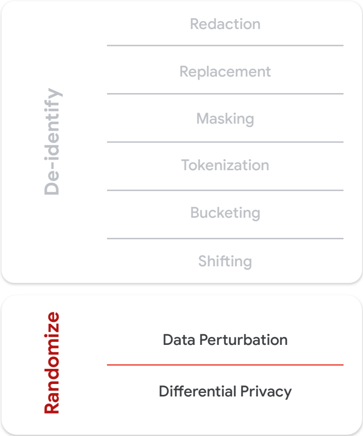
Choose:

- **Data Perturbation** for ease-of-implementation.
- **Differential Privacy (DP)** for more rigorous stronger privacy guarantee.

Whereas differential privacy provides a more rigorous privacy guarantee.

Let’s explore each of these techniques by looking at each method.

Data security methods for privacy in ML

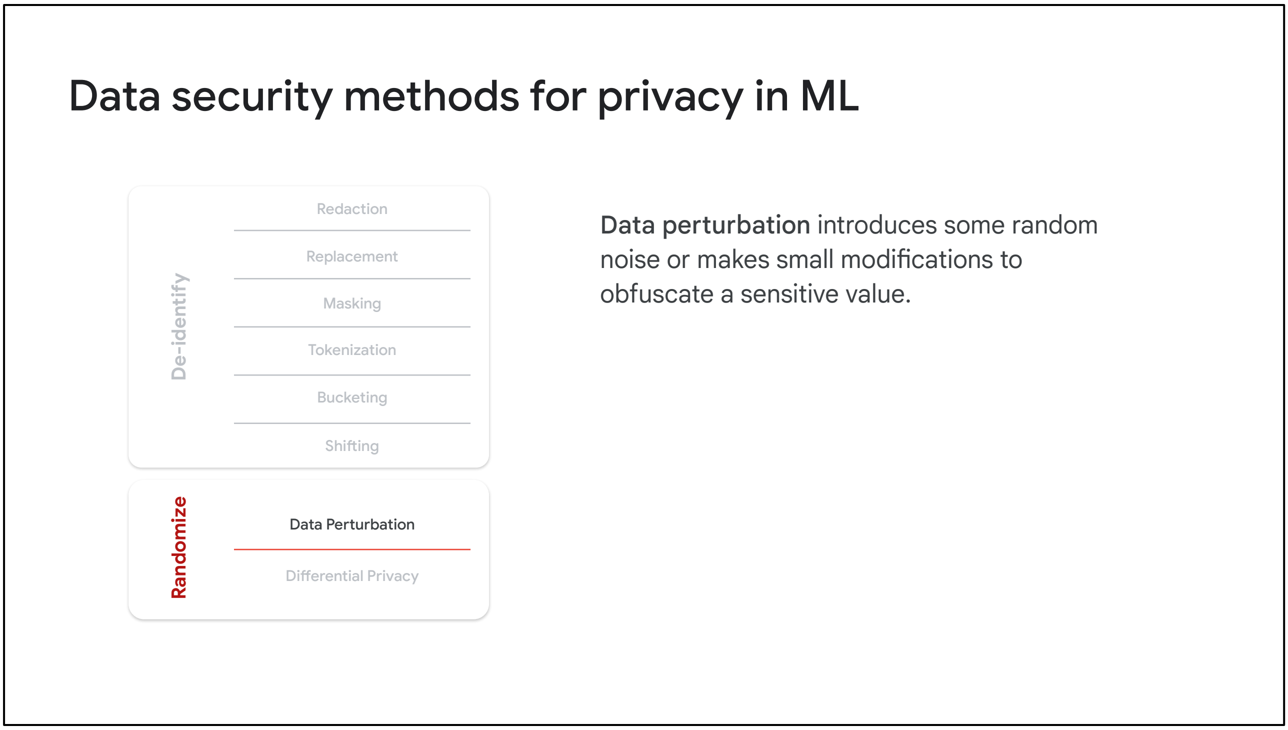


De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy



Data security methods for privacy in ML

De-identify

Redaction

Replacement

Masking

Tokenization

Bucketing

Shifting

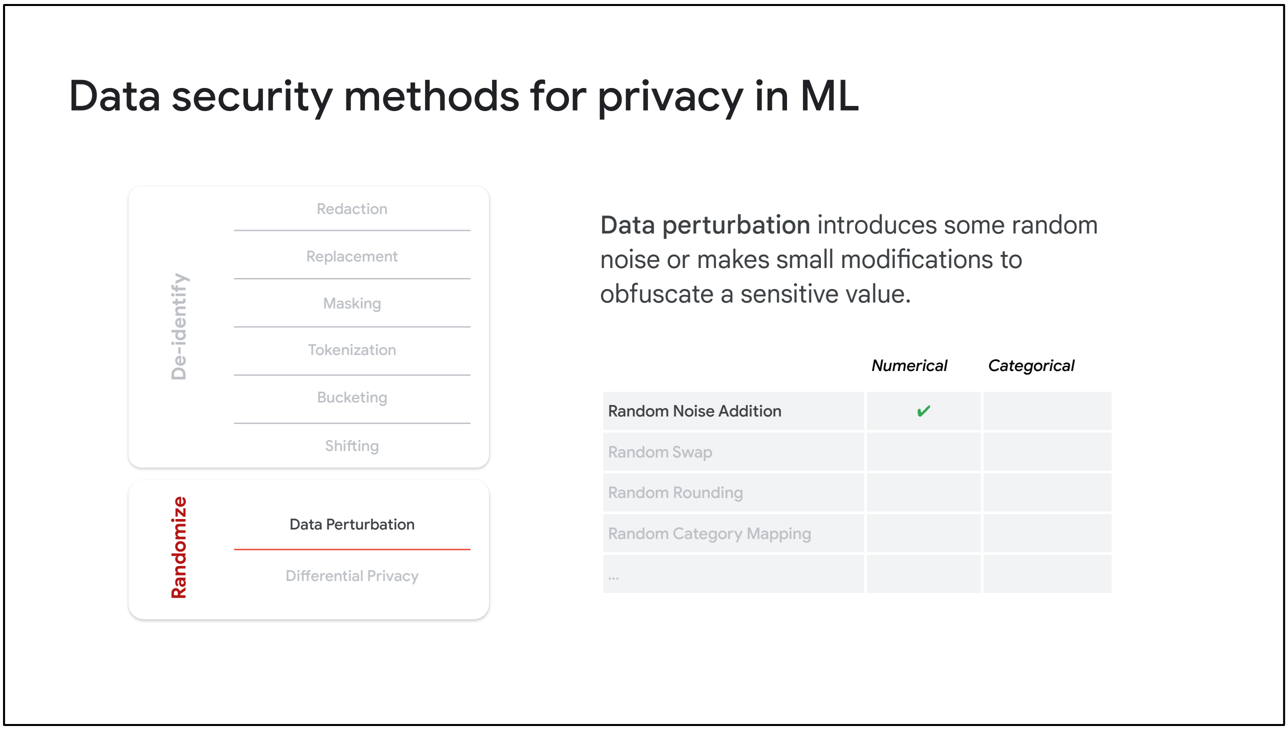
Randomize

Data Perturbation

Differential Privacy

Data perturbation introduces some random noise or makes small modifications to obfuscate a sensitive value.

Data perturbation introduces some random noise or makes small modifications to obfuscate a sensitive value.
The perturbed data still reflects the general trends and patterns present in the original dataset while preventing the direct identification of individuals.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

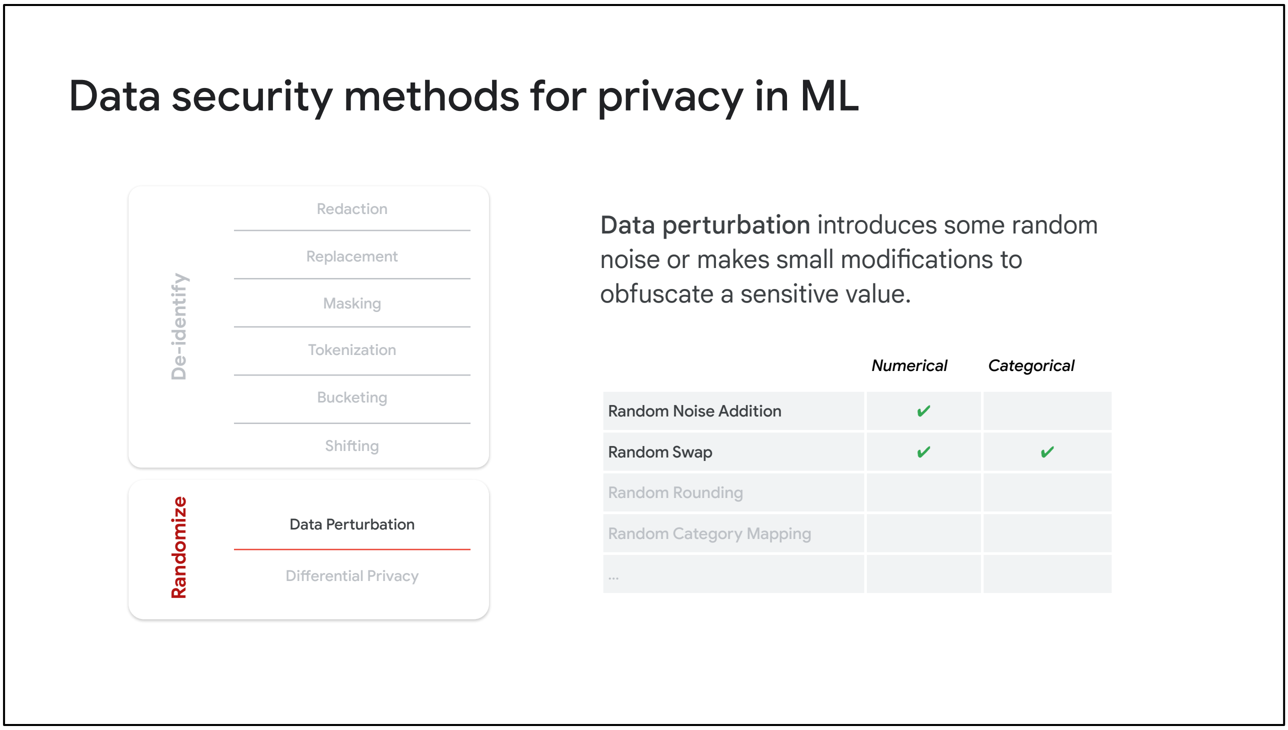
- Data Perturbation
- Differential Privacy

Data perturbation introduces some random noise or makes small modifications to obfuscate a sensitive value.

	Numerical	*Categorical*
Random Noise Addition	✓	
Random Swap		
Random Rounding		
Random Category Mapping		
...		

Some examples of data perturbation techniques include:

- Random noise addition: adding random values from a small range to each data point in the dataset. This works for numerical values.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

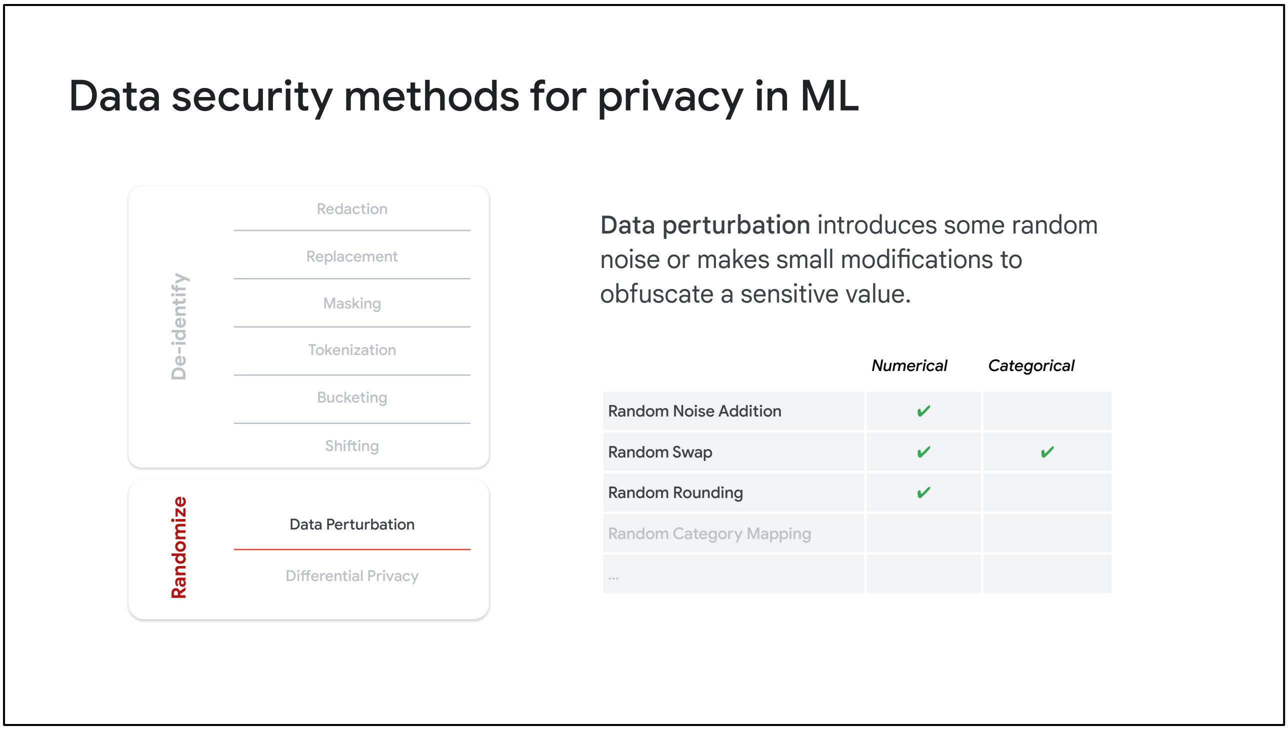
Randomize

- Data Perturbation
- Differential Privacy

Data perturbation introduces some random noise or makes small modifications to obfuscate a sensitive value.

	Numerical	*Categorical*
Random Noise Addition	✓	
Random Swap	✓	✓
Random Rounding		
Random Category Mapping		
...		

- Random swap: exchanging the values of different data points randomly.This works for both numerical and categorical values.



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

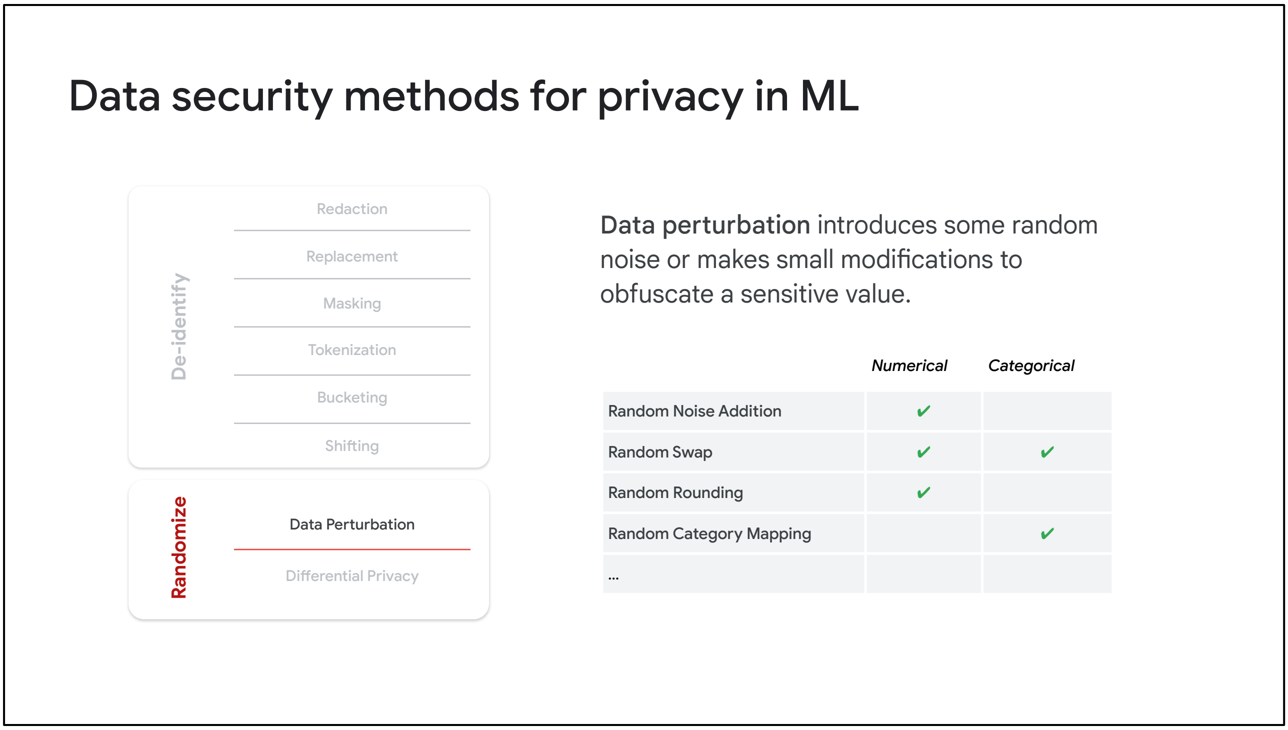
Randomize

- Data Perturbation
- Differential Privacy

Data perturbation introduces some random noise or makes small modifications to obfuscate a sensitive value.

	Numerical	*Categorical*
Random Noise Addition	✓	
Random Swap	✓	✓
Random Rounding	✓	
Random Category Mapping		
...		

- Random rounding: rounding data points to random precision levels. This works for numerical values.



Data security methods for privacy in ML

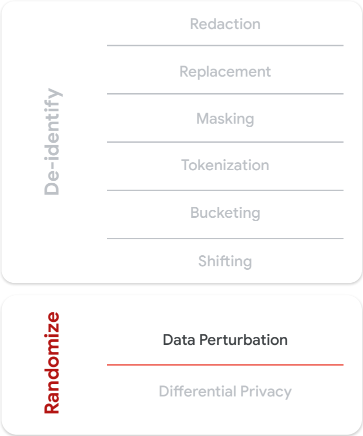
Data perturbation introduces some random noise or makes small modifications to obfuscate a sensitive value.

	Numerical	*Categorical*
Random Noise Addition	✓	
Random Swap	✓	✓
Random Rounding	✓	
Random Category Mapping		✓
...		

- Random category mapping: randomly mapping categorical values to different categories within the same variable.

The level of perturbation must strike a balance between privacy protection and data utility, ensuring that the data remains informative and valuable for legitimate analytical purposes.

Data security methods for privacy in ML

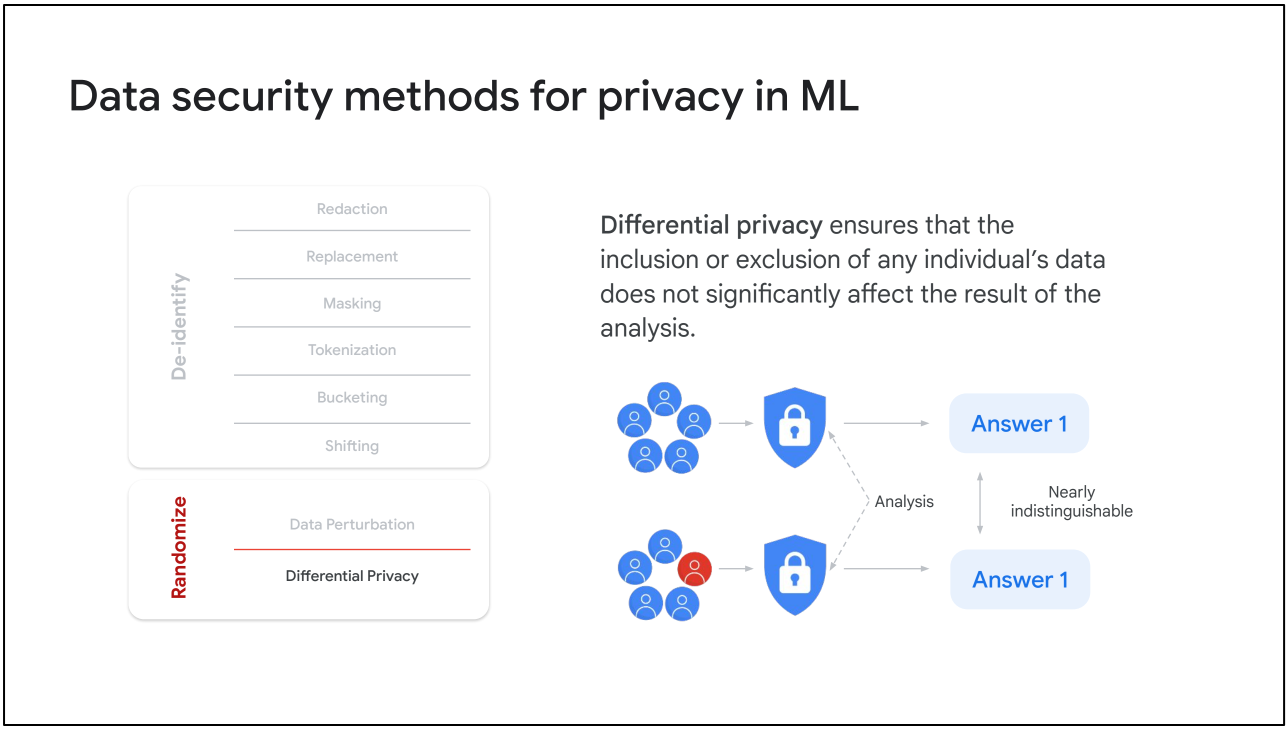


De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy



Data security methods for privacy in ML

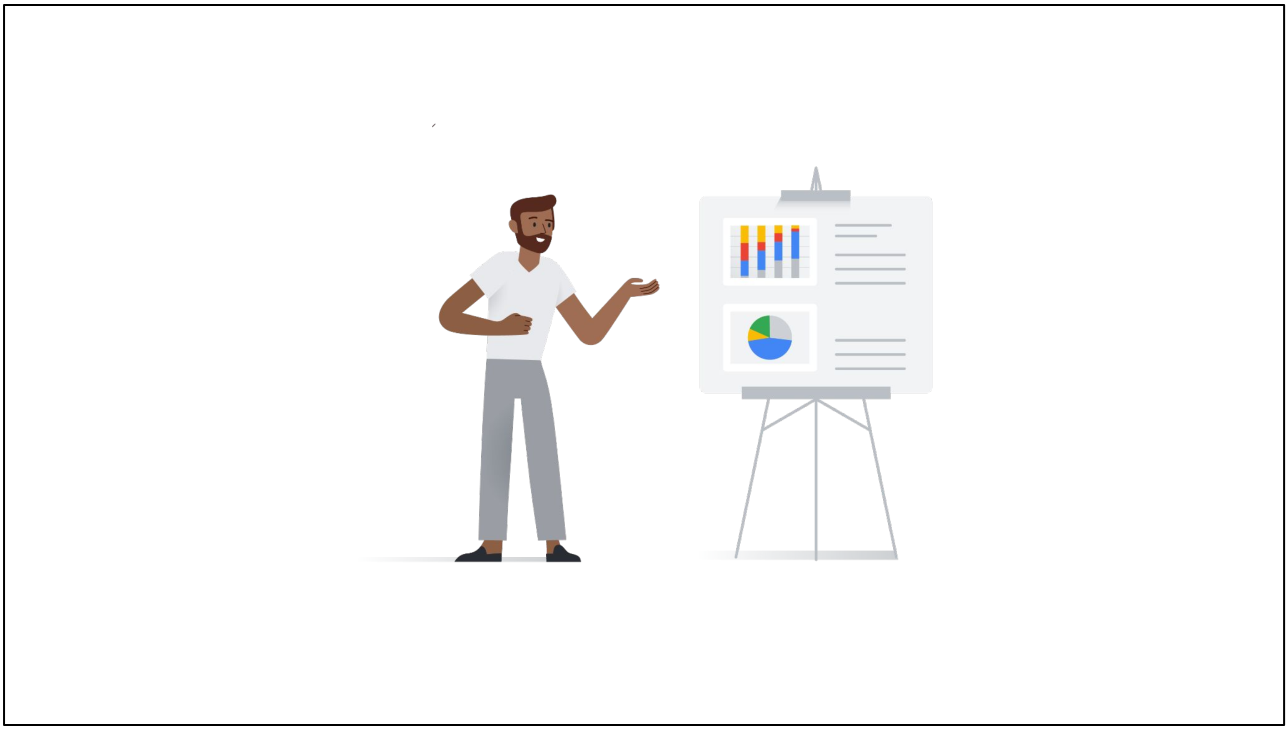
De-identify
- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize
- Data Perturbation
- Differential Privacy

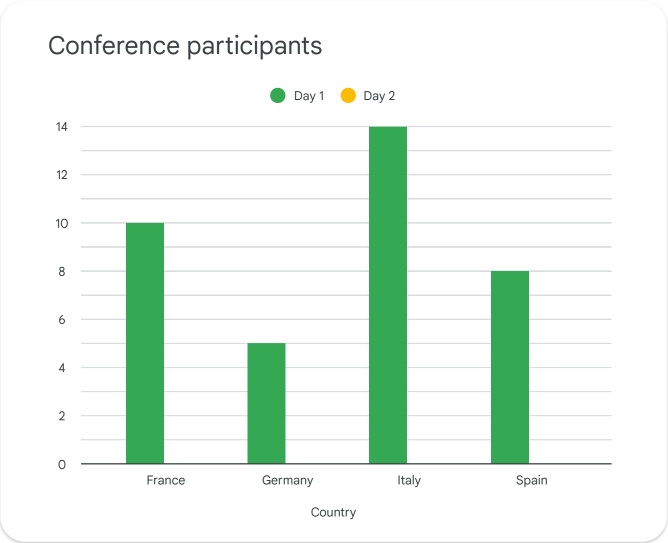
Differential privacy ensures that the inclusion or exclusion of any individual's data does not significantly affect the result of the analysis.

Differential privacy ensures that the inclusion or exclusion of any individual's data does not significantly affect the output or result of the analysis. This is particularly useful when you want to share data, or allow inferences about groups of people while preventing someone from learning information about an individual.

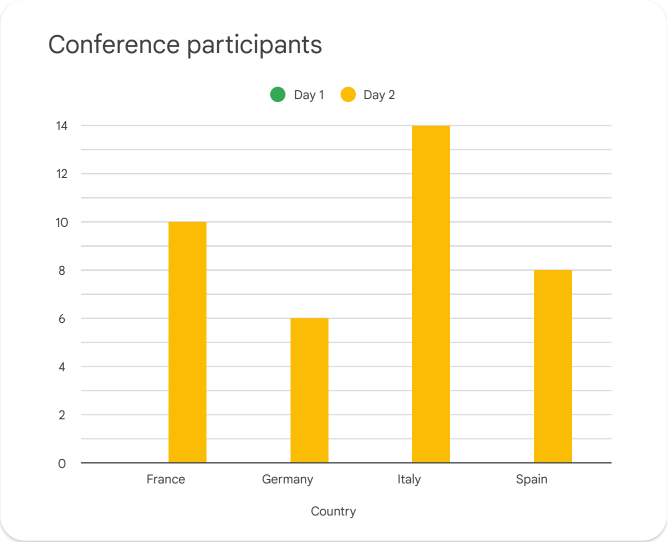
It is referred to as a "rigorous" approach since it provides strong, provable privacy guarantees.



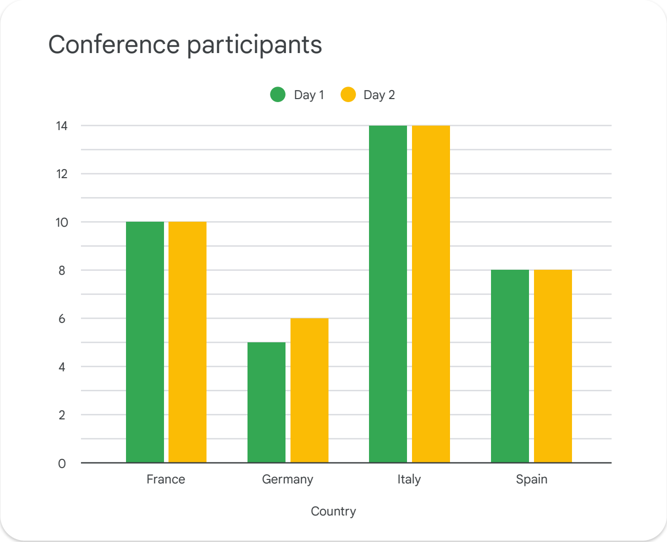
Let’s take a look at an example of applying differential privacy.
Imagine you are attending a conference and the moderator shares some insights about the audience.



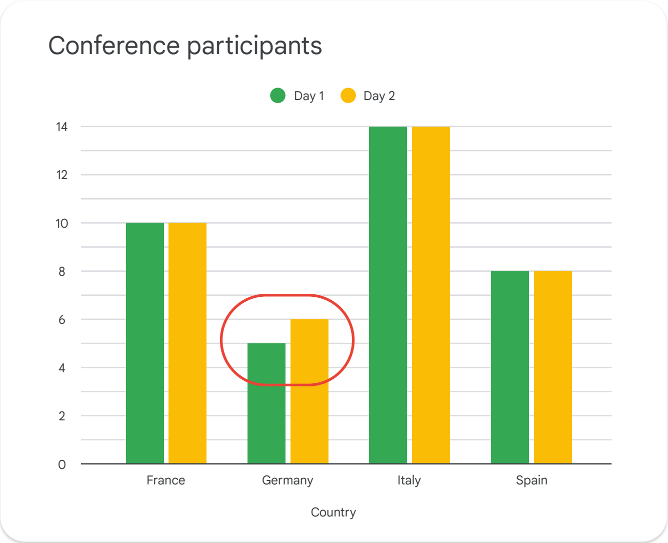
The moderator shows the country of origin of the participants in the room for day 1. The data is aggregated so it does not reveal information about any of the individuals in the room. The data reveals the top country of origin is Italy, whereas Germany has the least number of participants, and France and Spain participant numbers are in the middle.



On day 2, the moderator shows the stats for the day. It looks very similar to the chart from day 1.

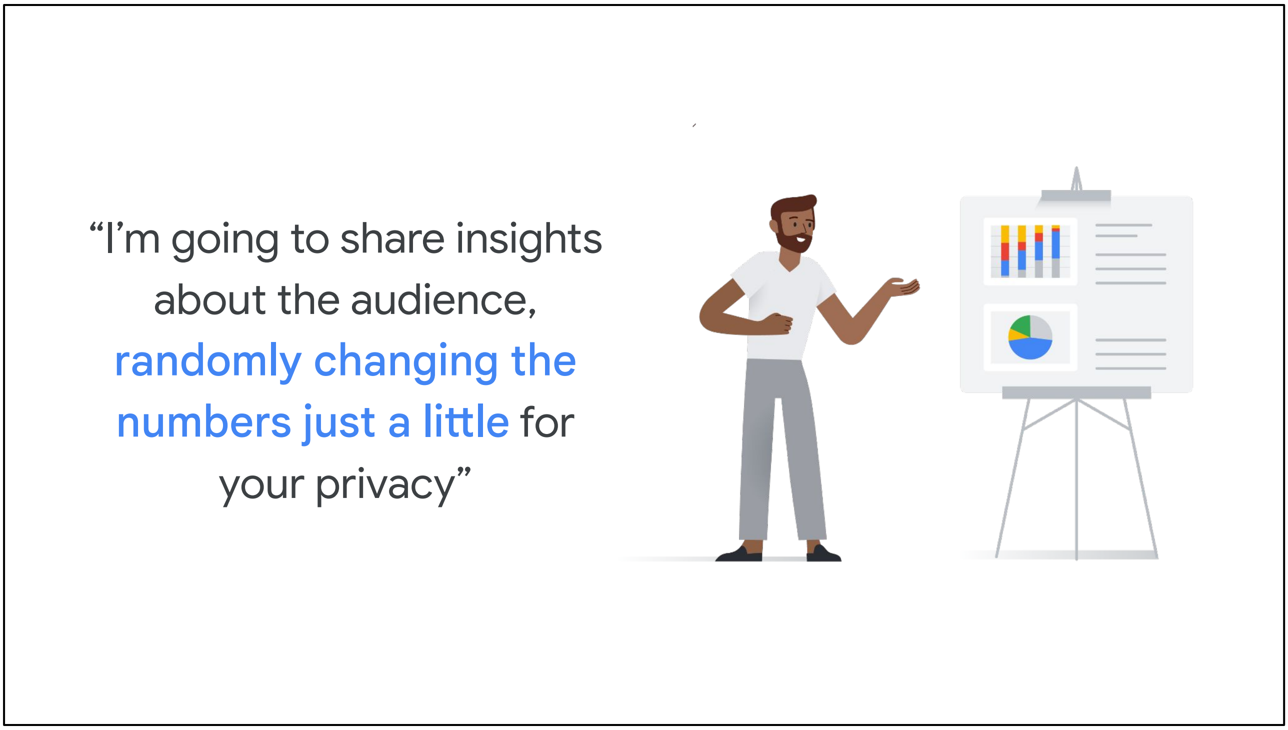


However, before the Day 2 session started, you met someone who attended Day 2, but not Day 1. Knowing this information, you compare the data from both days together. Are you able to determine the country of origin for the new attendee based on the data sets?



Yes, the new attendee is from Germany!

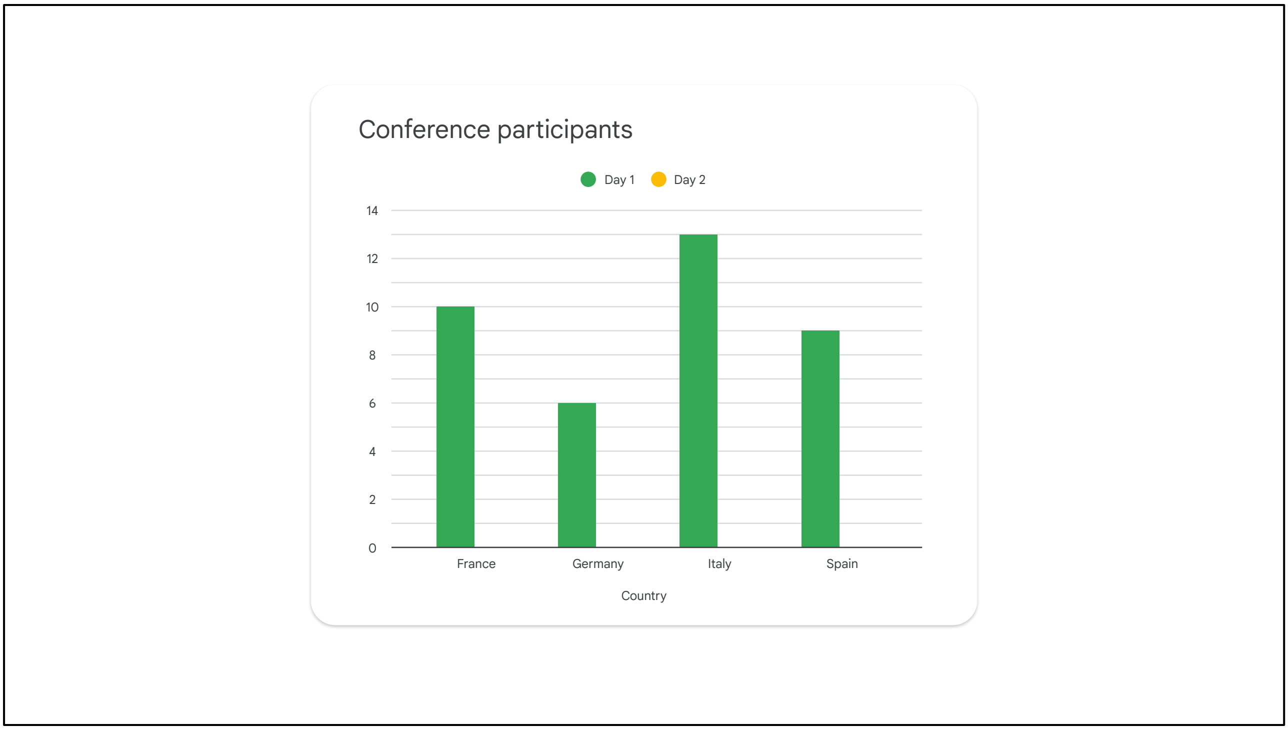
This is an example of how given two aggregated data sets, the moderator has not protected the privacy of a particular participant.



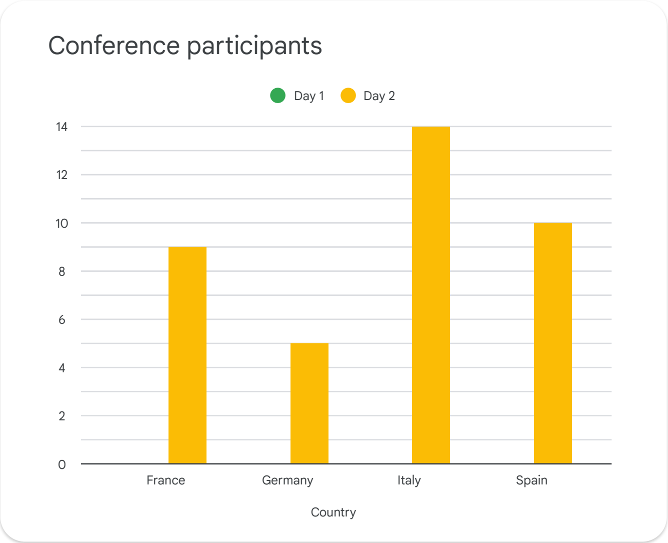
"I'm going to share insights about the audience, **randomly changing the numbers just a little** for your privacy"

To address the data privacy leak issue, the moderator needs a new approach.

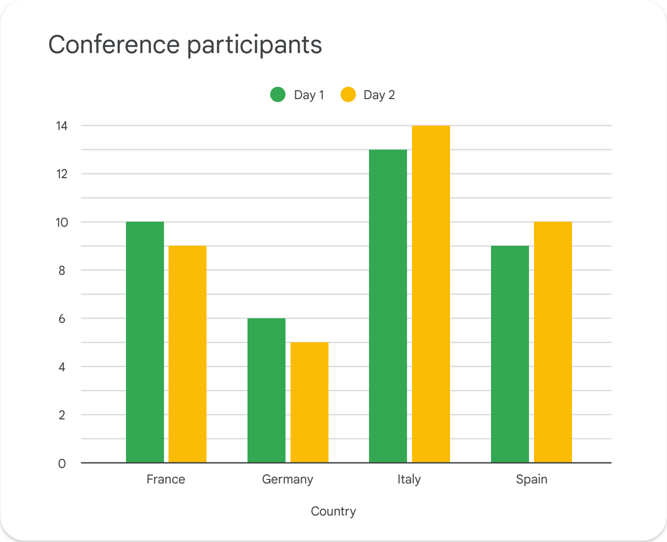
Let's try to randomly change the numbers a little in order to improve data privacy.



And now, look at how the day one stats look compared to the real numbers. They're slightly different. It is less precise, but you still get the same insights about the statistical trends.

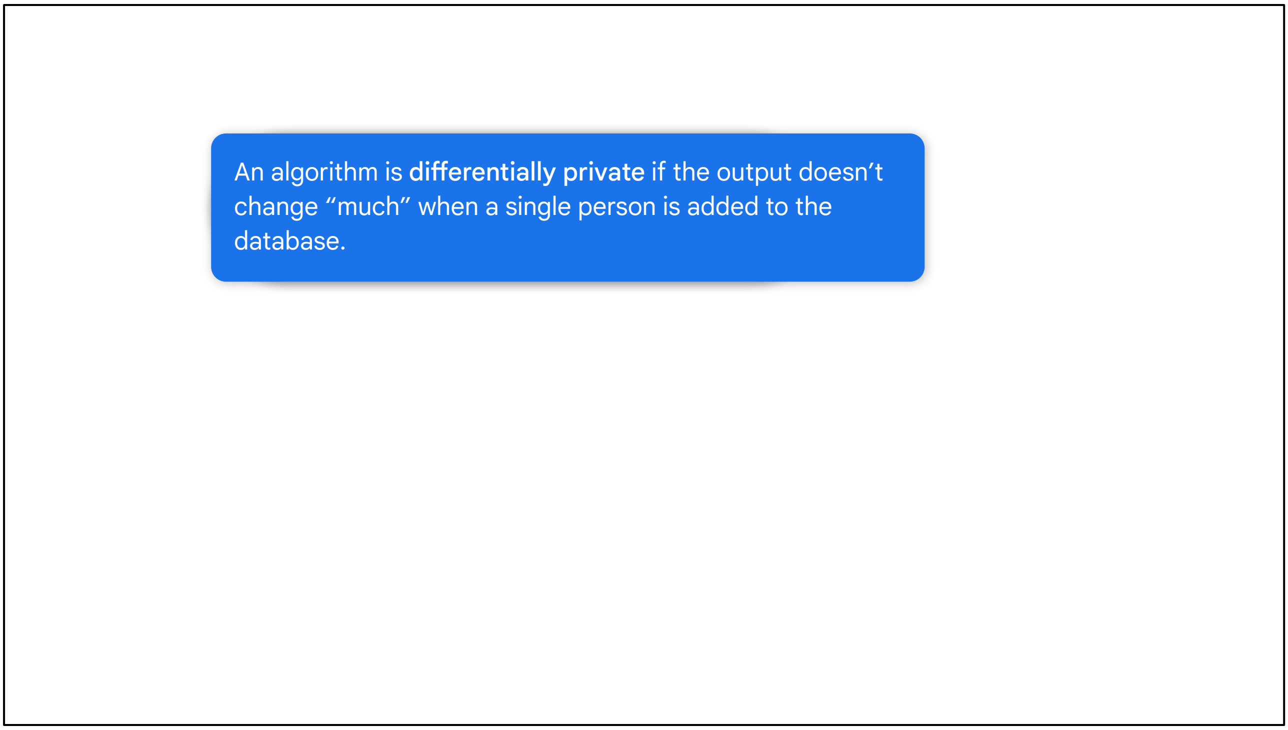


The same approach is applied to the data from Day 2.



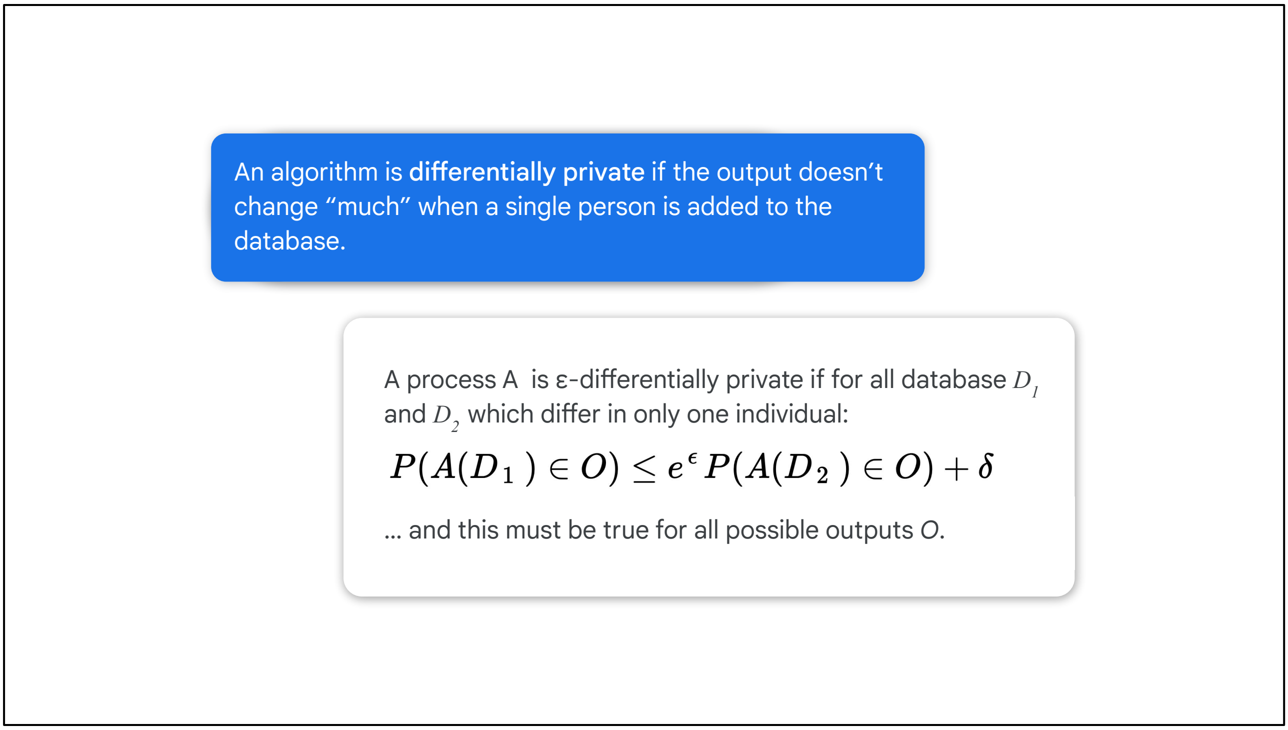
And now, if you compare the stats for day 1 and day 2, you can’t tell which country the single new attendee is coming from, and it’s not worth trying to figure out since you know the numbers were changed a little.

This is the principle behind differential privacy. Differential privacy introduces the right amount of noise to the data, so we’re able to provide useful, statistically significant insights, while ensuring that no one can tell if a particular individual's information was used in the computation. The best thing about this principle is that it is mathematically proven.



An algorithm is **differentially private** if the output doesn’t change “much” when a single person is added to the database.

Differential privacy relies on this equation. It essentially says that something is differentially private if the output doesn’t change “much” when a single person is added to the database.



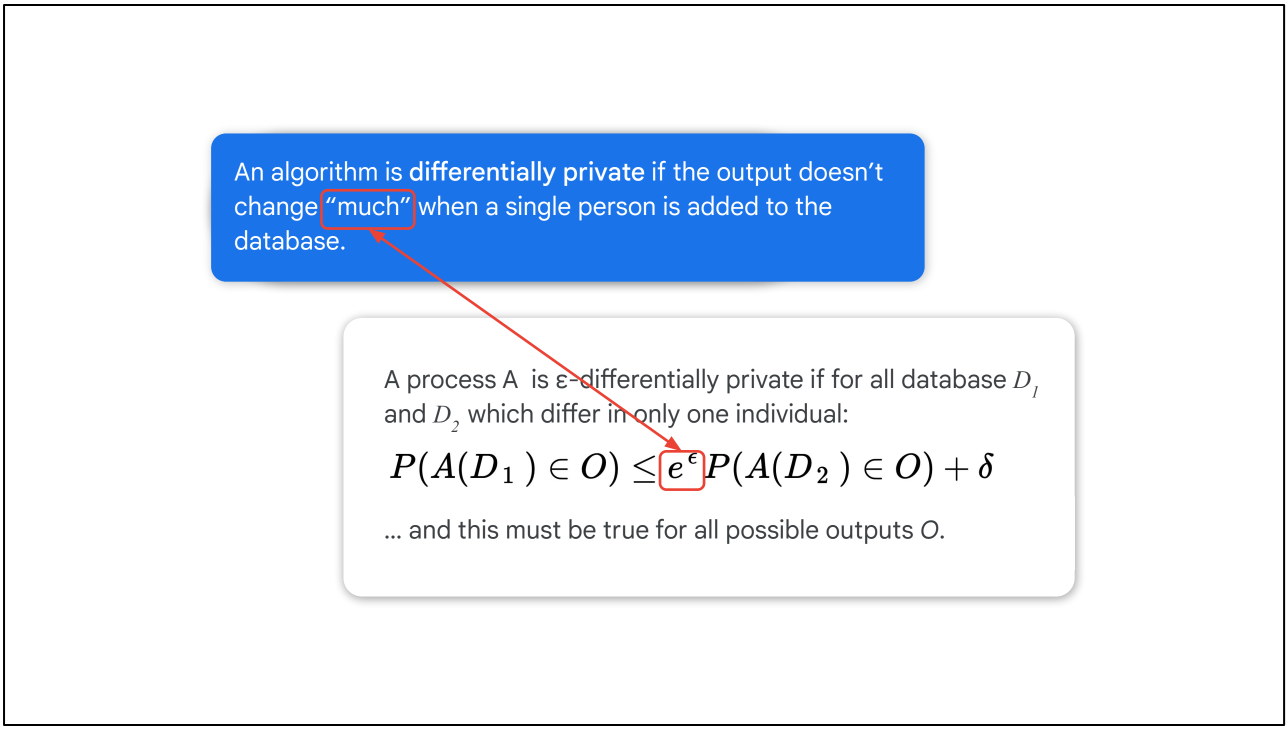
An algorithm is **differentially private** if the output doesn’t change “much” when a single person is added to the database.

A process A is ε-differentially private if for all database D_1 and D_2 which differ in only one individual:

$$P(A(D_1) \in O) \leq e^{\epsilon} P(A(D_2) \in O) + \delta$$

... and this must be true for all possible outputs *O*.

Going back to the previous example,



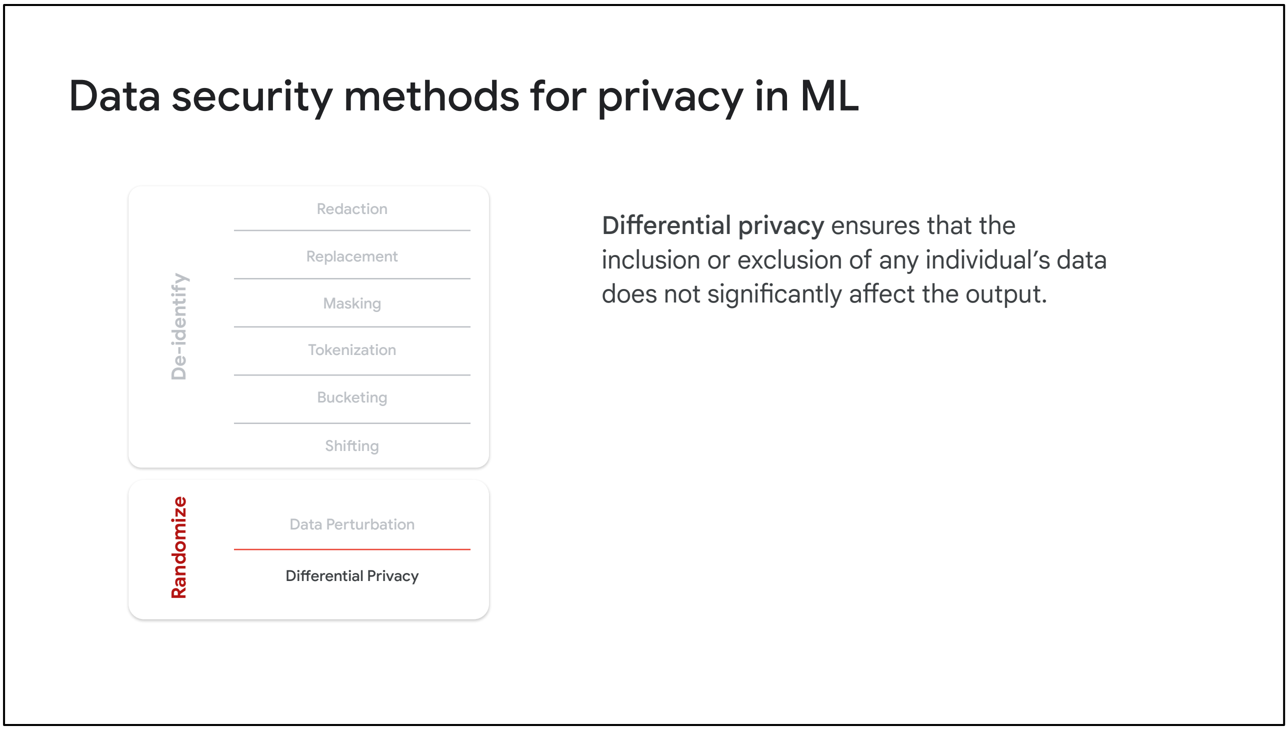
An algorithm is **differentially private** if the output doesn’t change ”much” when a single person is added to the database.

A process A is ε-differentially private if for all database D_1 and D_2 which differ in only one individual:

$$P(A(D_1) \in O) \leq \boxed{e^{\epsilon}} P(A(D_2) \in O) + \delta$$

… and this must be true for all possible outputs *O*.

a differentially private version of those charts doesn’t allow us to tell much about the participants when a new one is added.



Data security methods for privacy in ML

De-identify

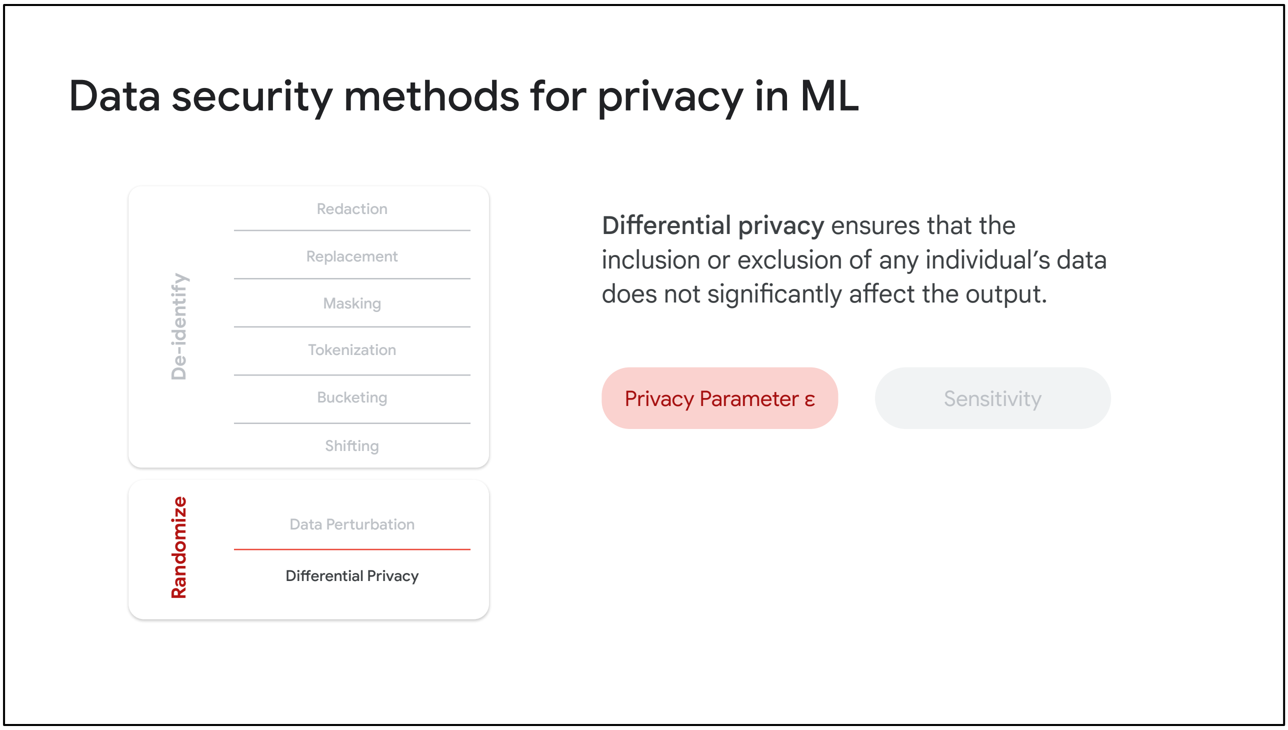
- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

Differential privacy ensures that the inclusion or exclusion of any individual's data does not significantly affect the output.

Differential privacy is characterized by two key parameters:



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

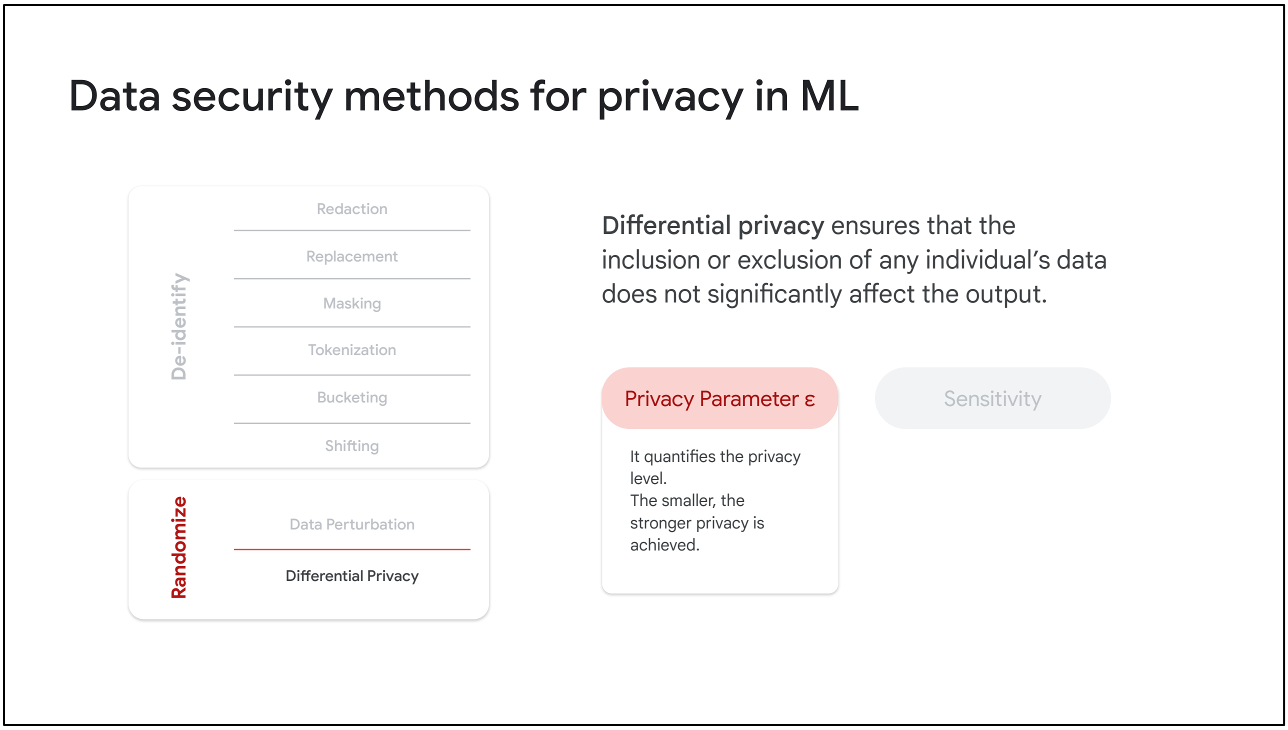
- Data Perturbation
- Differential Privacy

Differential privacy ensures that the inclusion or exclusion of any individual's data does not significantly affect the output.

Privacy Parameter ε

Sensitivity

- The privacy parameter ε (epsilon)



Data security methods for privacy in ML

De-identify

- Redaction
- Replacement
- Masking
- Tokenization
- Bucketing
- Shifting

Randomize

- Data Perturbation
- Differential Privacy

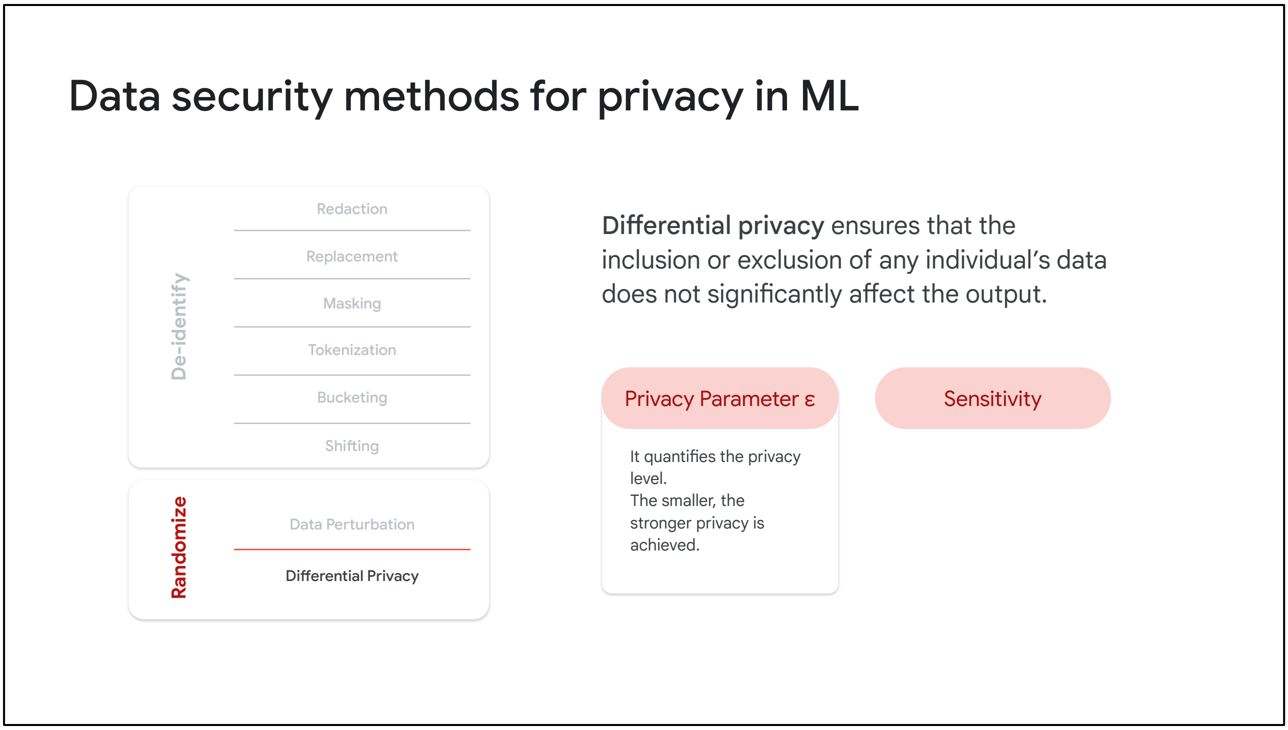
Differential privacy ensures that the inclusion or exclusion of any individual's data does not significantly affect the output.

Privacy Parameter ε

It quantifies the privacy level.
The smaller, the stronger privacy is achieved.

Sensitivity

is a non-negative value that quantifies the privacy level provided by the differential privacy mechanism. A smaller ε value indicates stronger privacy protection; but it comes with the risk of too much noise being added to the query results, potentially reducing data utility.



Data security methods for privacy in ML

De-identify

Redaction

Replacement

Masking

Tokenization

Bucketing

Shifting

Randomize

Data Perturbation

Differential Privacy

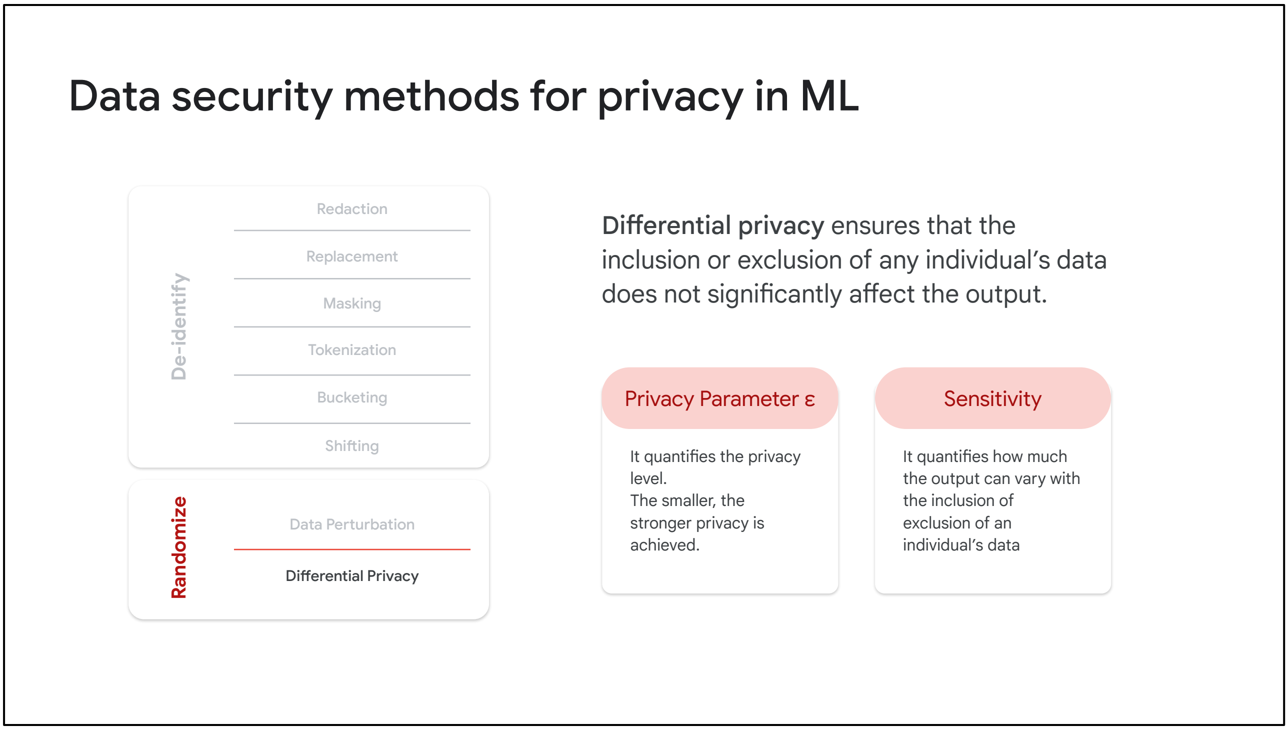
Differential privacy ensures that the inclusion or exclusion of any individual’s data does not significantly affect the output.

Privacy Parameter ε

It quantifies the privacy level.
The smaller, the stronger privacy is achieved.

Sensitivity

- Sensitivity measures



Data security methods for privacy in ML

Differential privacy ensures that the inclusion or exclusion of any individual's data does not significantly affect the output.

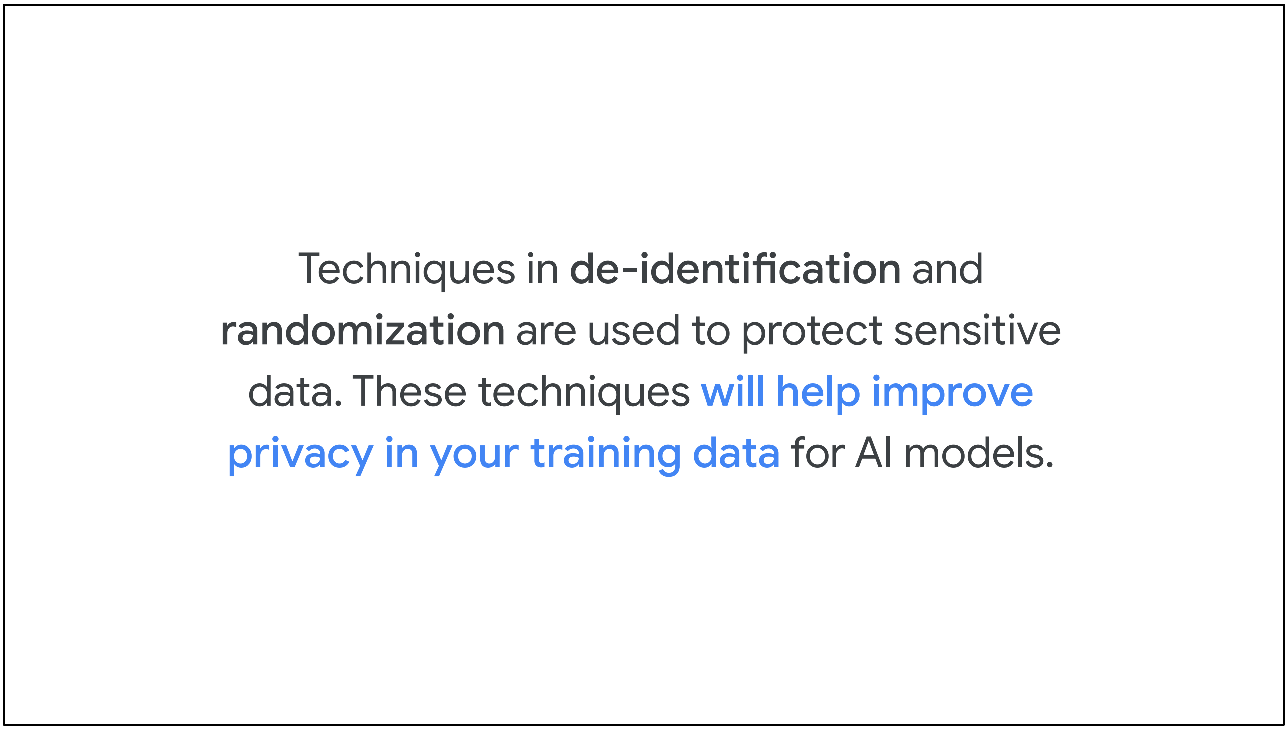
Privacy Parameter ε

It quantifies the privacy level.
The smaller, the stronger privacy is achieved.

Sensitivity

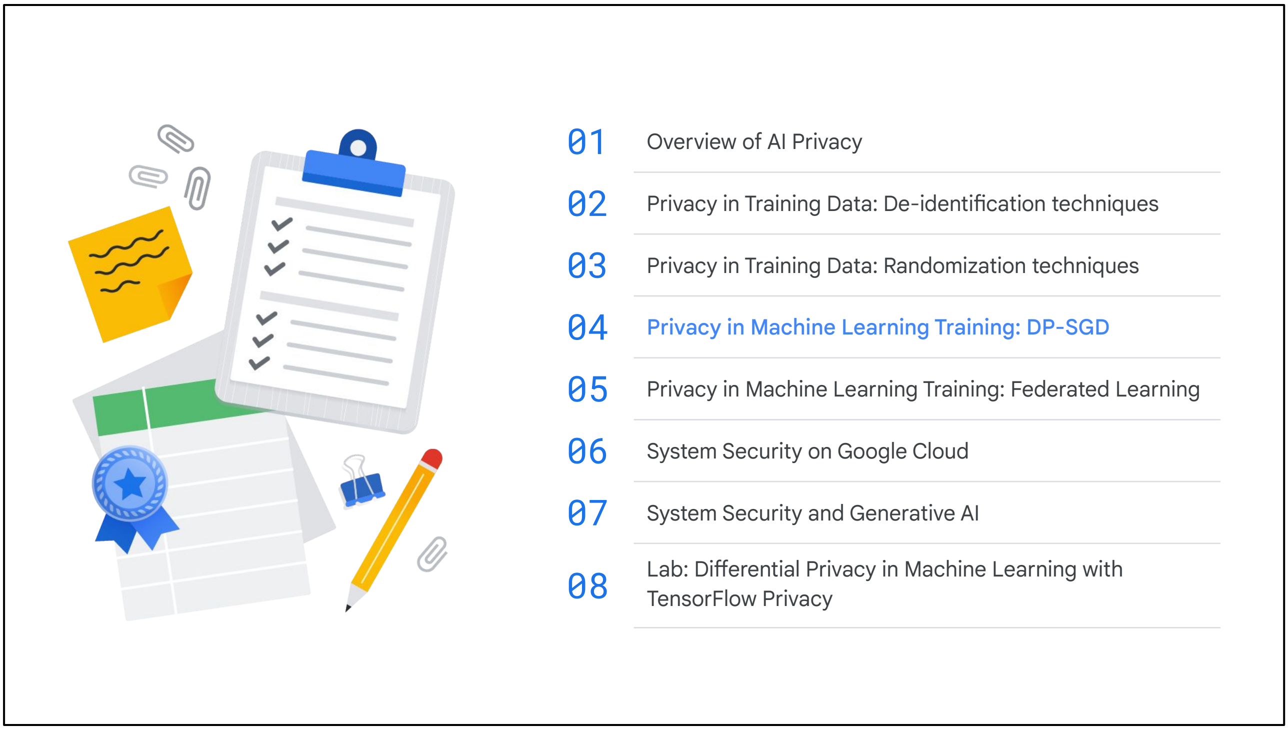
It quantifies how much the output can vary with the inclusion of exclusion of an individual's data

how much the output of a query can change, specifically the maximum absolute difference, when a single individual's data is added or removed from the dataset. Sensitivity is crucial for determining the amount of noise that needs to be added to ensure privacy.



Techniques in **de-identification** and **randomization** are used to protect sensitive data. These techniques will help improve privacy in your training data for AI models.

Techniques in de-identification and randomization are used to protect sensitive data. These techniques will help improve privacy in your training data for AI models.



01 Overview of AI Privacy

02 Privacy in Training Data: De-identification techniques

03 Privacy in Training Data: Randomization techniques

04 Privacy in Machine Learning Training: DP-SGD

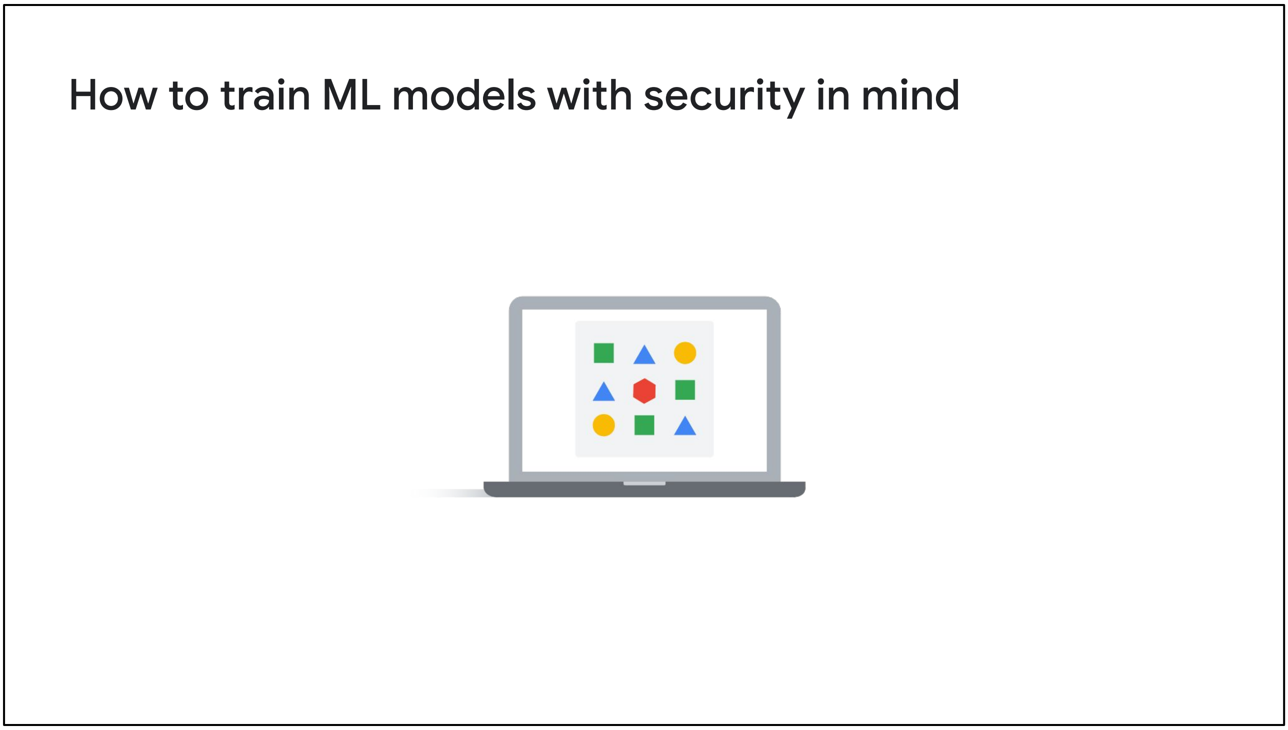
05 Privacy in Machine Learning Training: Federated Learning

06 System Security on Google Cloud

07 System Security and Generative AI

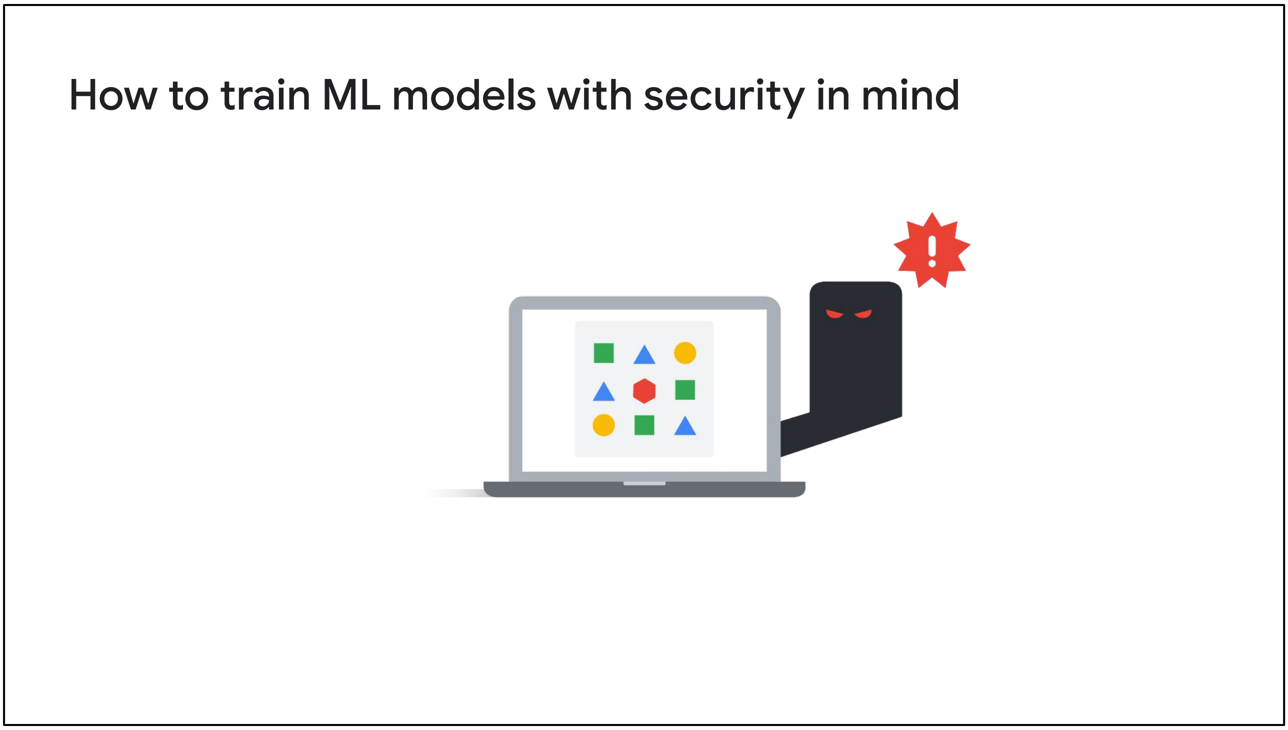
08 Lab: Differential Privacy in Machine Learning with TensorFlow Privacy

Let’s talk about privacy in machine learning training.



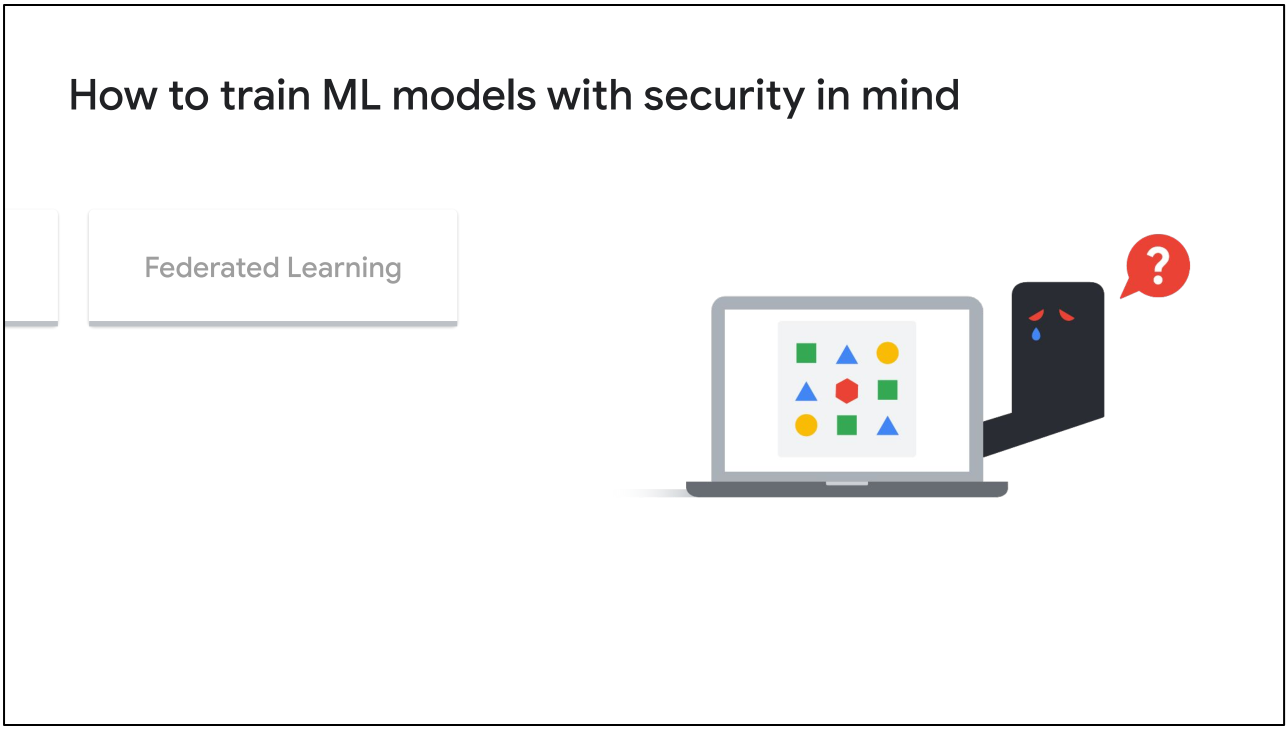
How to train ML models with security in mind

After building machine learning models, it is crucial to carefully consider security measures before exposing them to users,



How to train ML models with security in mind

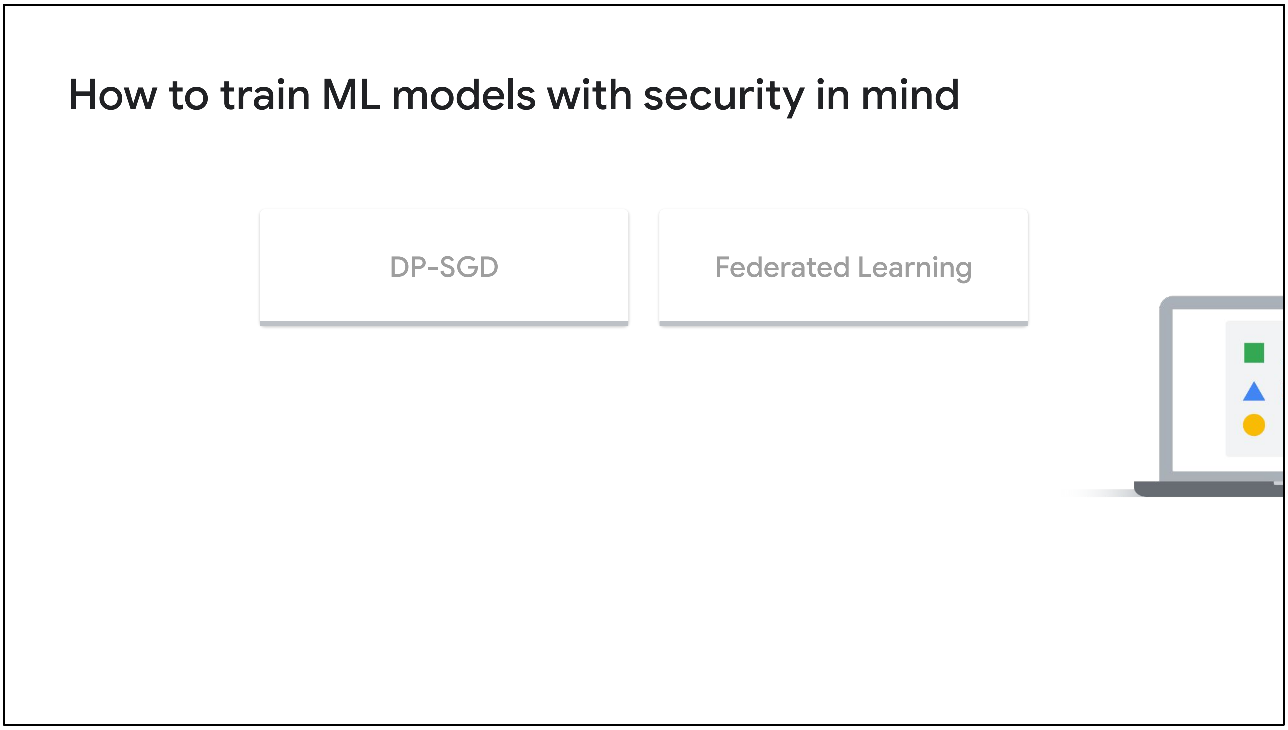
as malicious individuals may attempt various attacks to extract sensitive data or exploit model vulnerabilities.



How to train ML models with security in mind

Federated Learning

There are many types of attacks, and many measures we can take to safeguard against them.

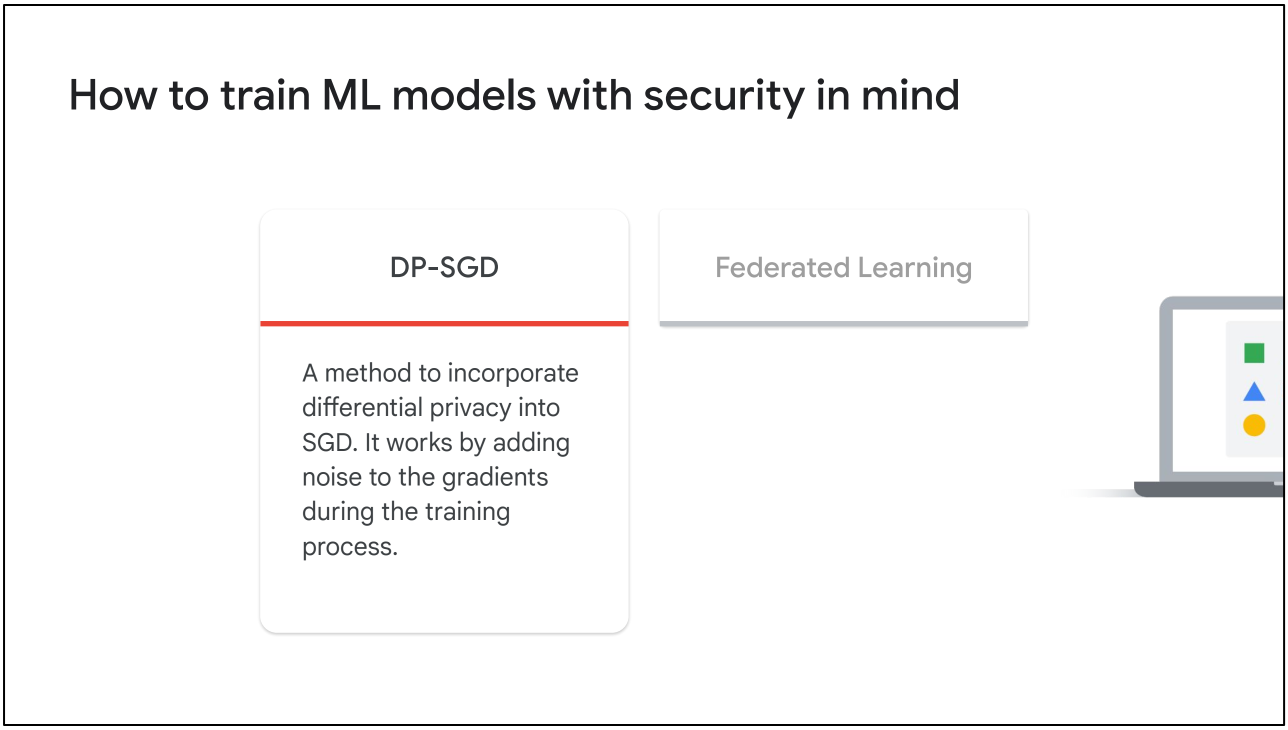


How to train ML models with security in mind

DP-SGD

Federated Learning

While there are many methods, two popular ones for secure models are:



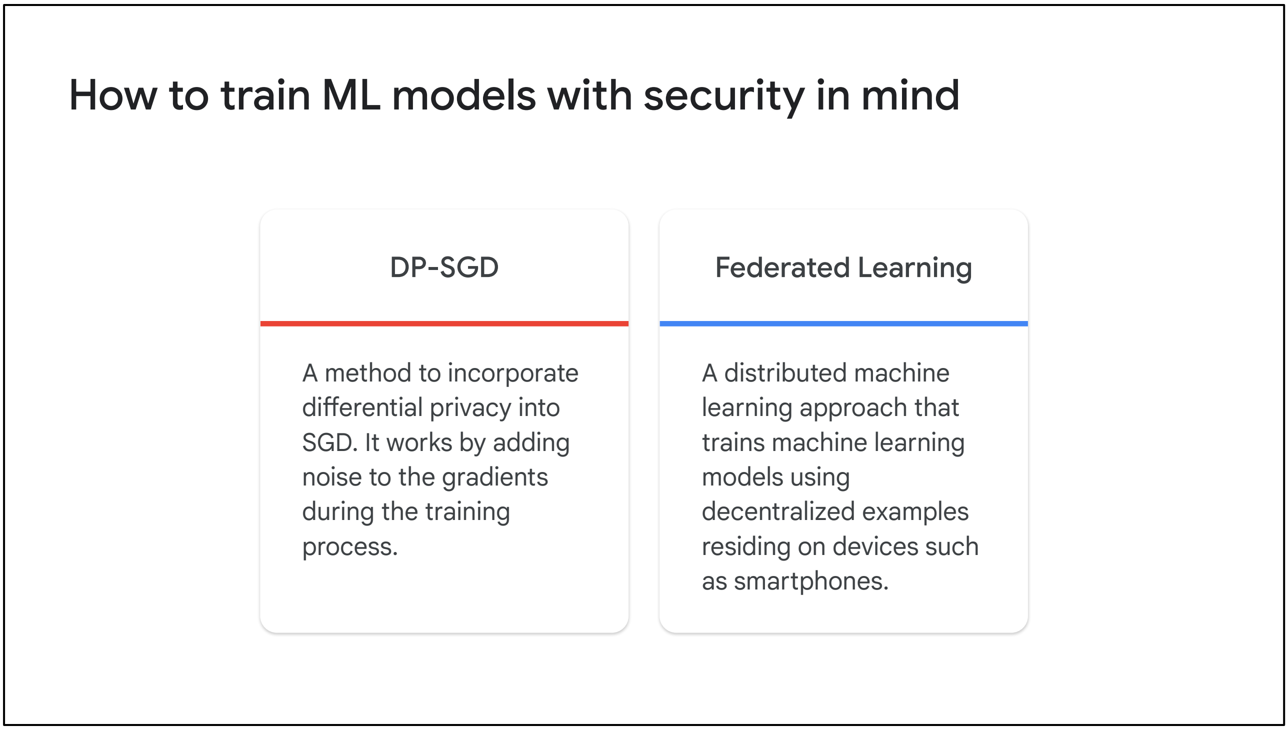
How to train ML models with security in mind

DP-SGD

A method to incorporate differential privacy into SGD. It works by adding noise to the gradients during the training process.

Federated Learning

Differentially Private Stochastic Gradient Descent, or DP-SGD,



How to train ML models with security in mind

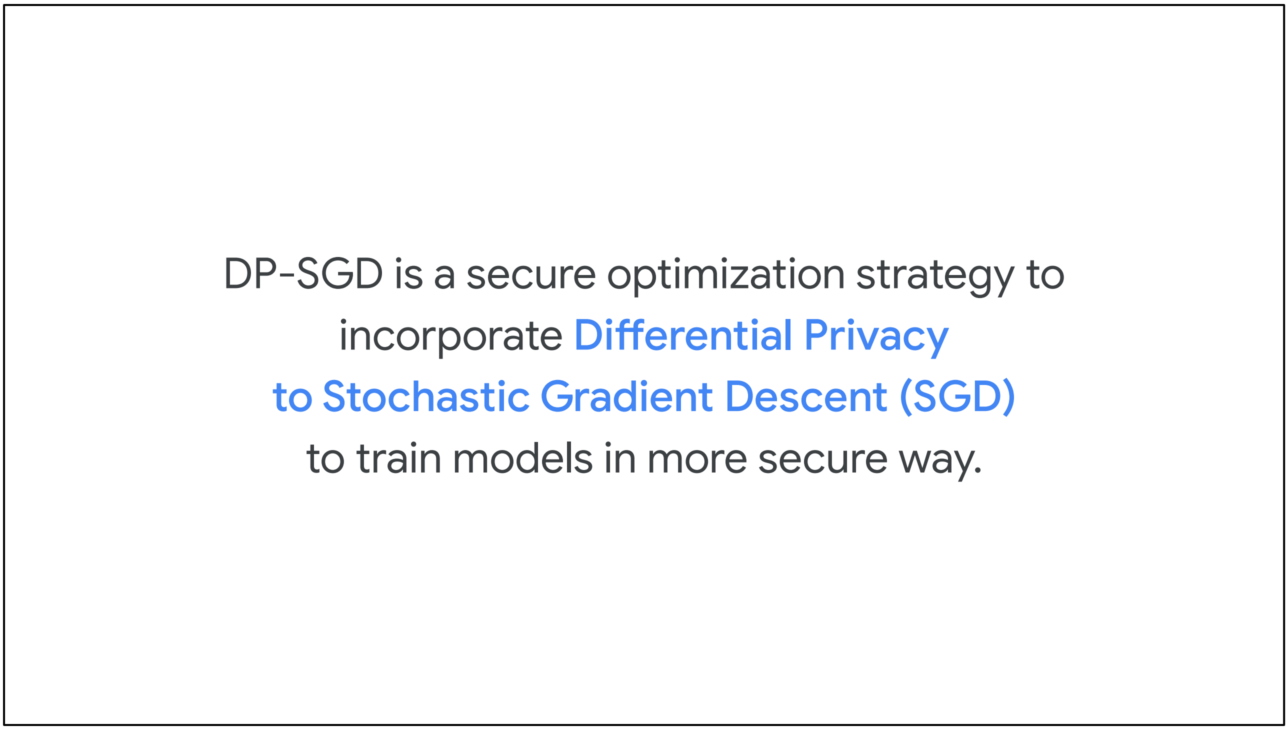
DP-SGD

A method to incorporate differential privacy into SGD. It works by adding noise to the gradients during the training process.

Federated Learning

A distributed machine learning approach that trains machine learning models using decentralized examples residing on devices such as smartphones.

and Federated Learning.

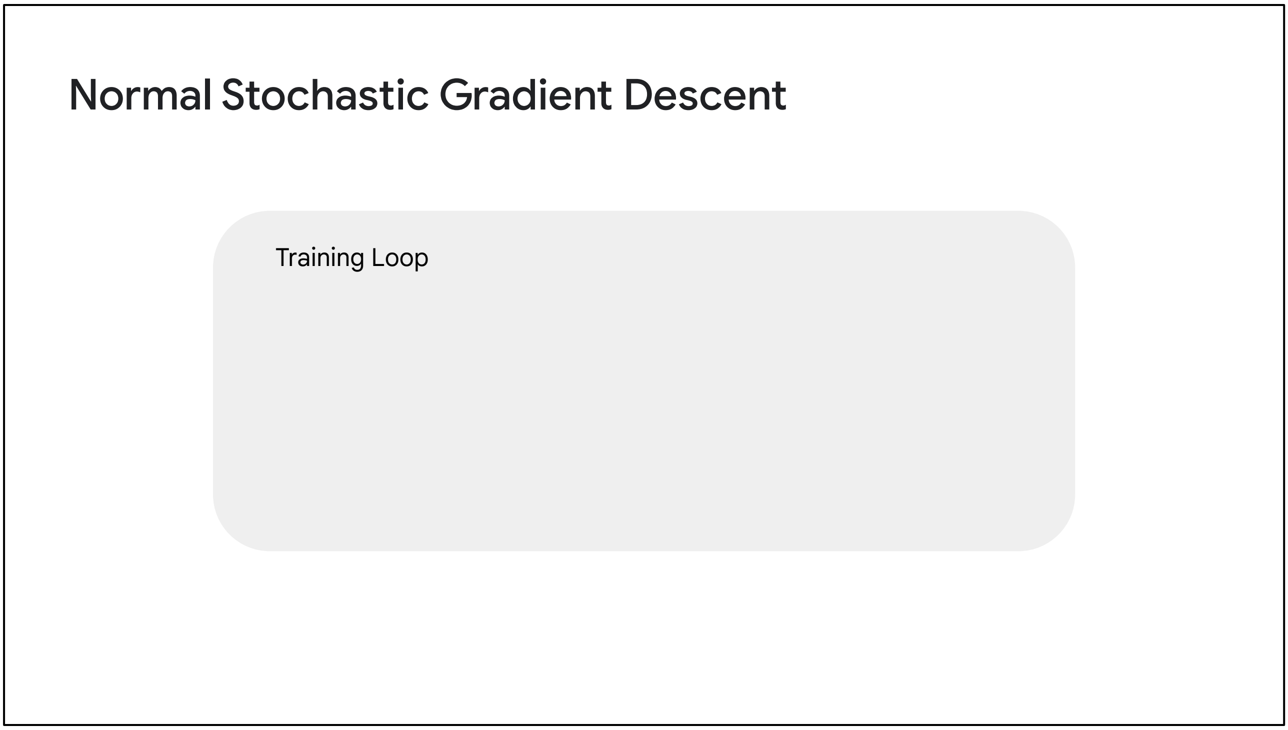


DP-SGD is a secure optimization strategy to incorporate **Differential Privacy to Stochastic Gradient Descent (SGD)** to train models in more secure way.

DP-SGD is a secure optimization strategy that incorporates differential privacy into Stochastic Gradient Descent (or SGD) to train models in a more secure way.
As mentioned earlier, differential privacy can be used for many use cases where we aggregate information in a secure way.

Machine learning optimization strategies like SGD, which update model parameters by computing gradients of them with respect to a group of data samples, can be viewed as aggregation algorithms. This means differential privacy can be integrated into optimization algorithms (like SGD) to protect sensitive information during the training process."

Although there are multiple implementations, let’s take a look at the simplest DP-SGD algorithm here..

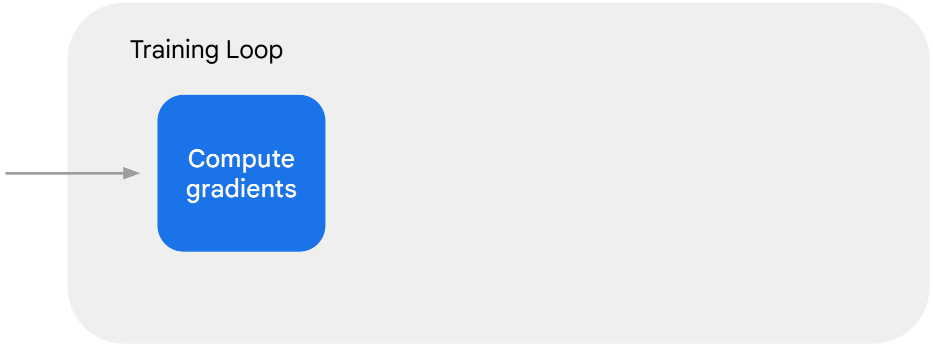


Normal Stochastic Gradient Descent

Training Loop

DP-SGD extends the normal SGD algorithm,

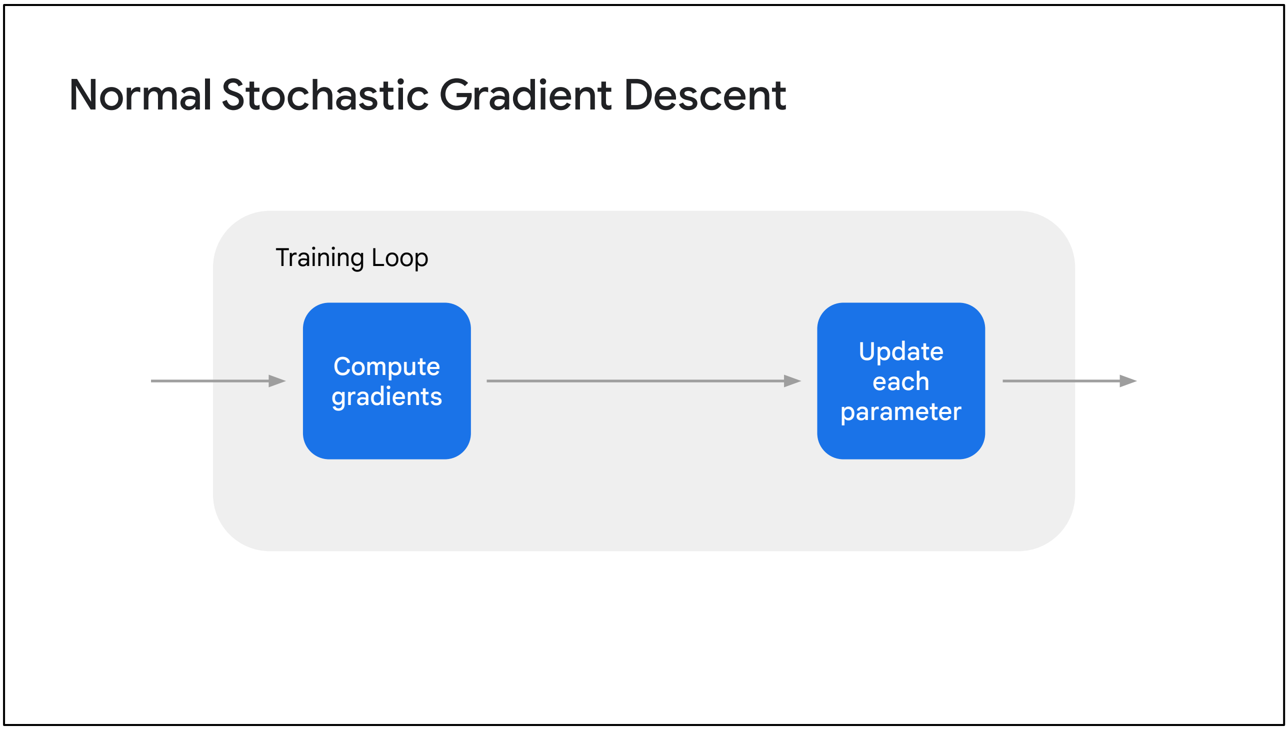
Normal Stochastic Gradient Descent



Training Loop

Compute gradients

which computes gradients of each parameter



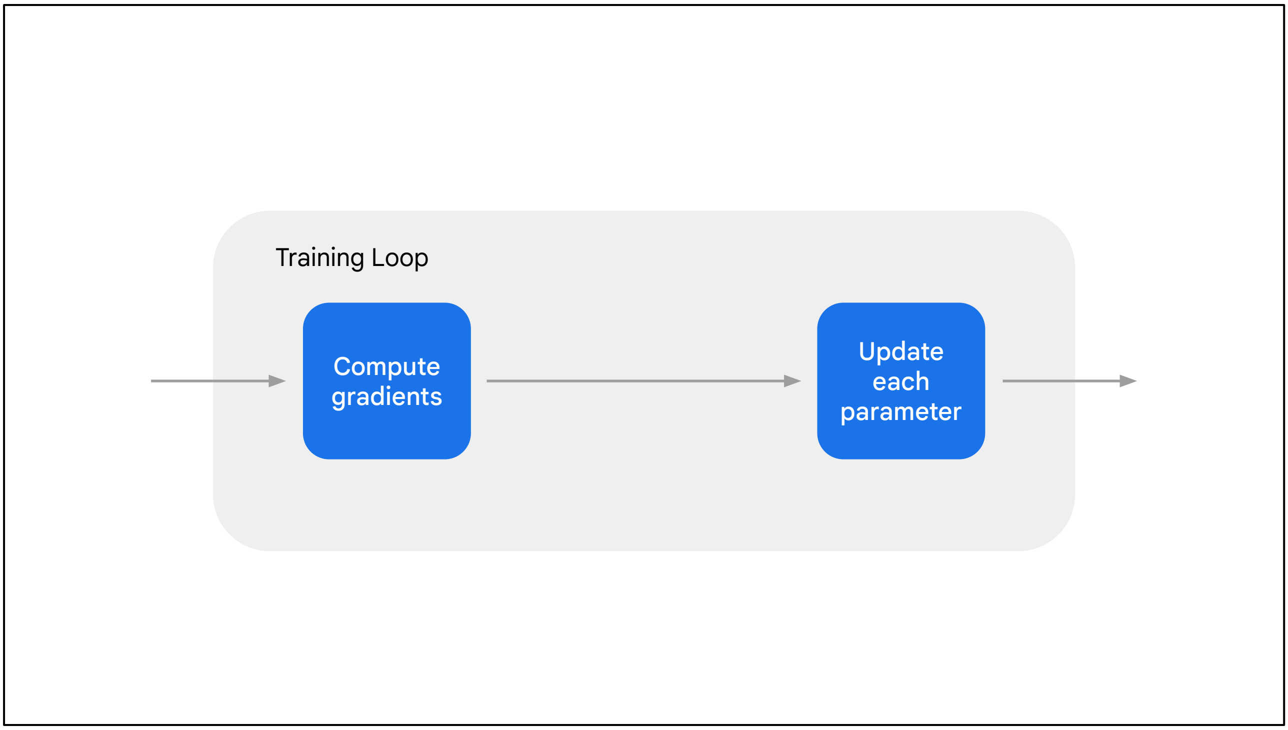
Normal Stochastic Gradient Descent

Training Loop

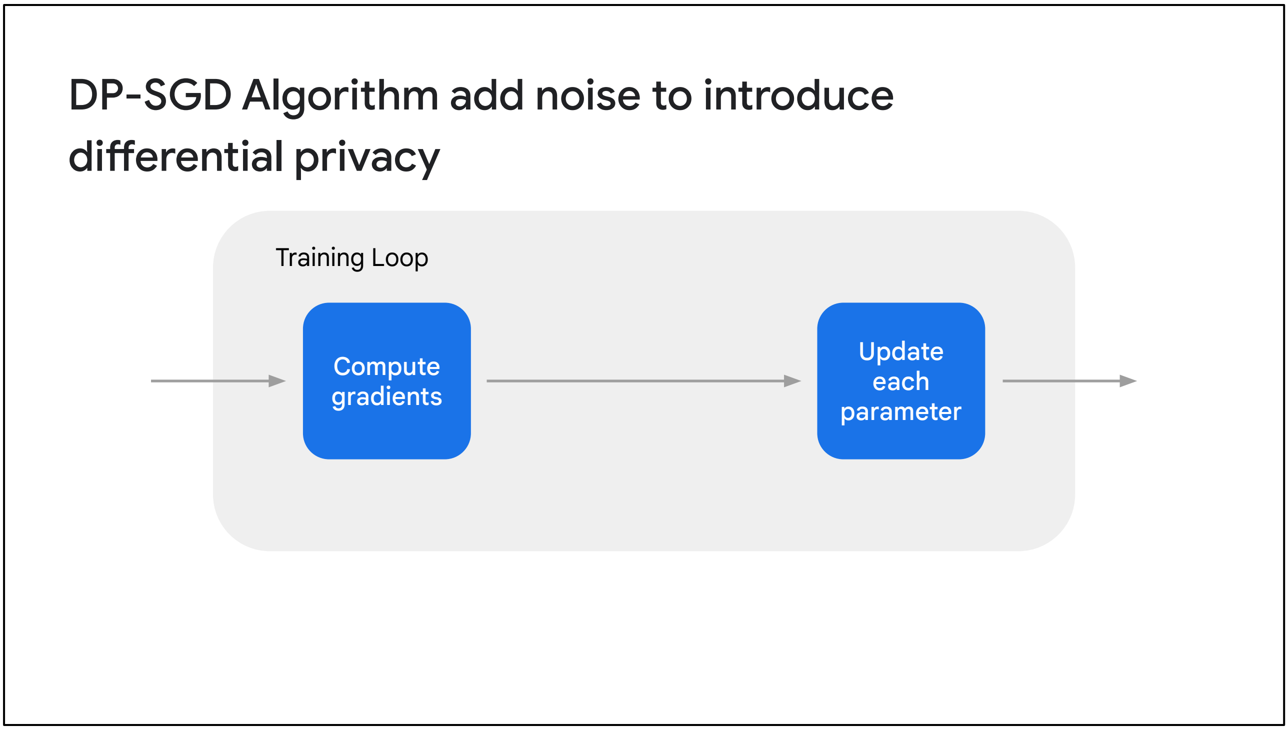
Compute gradients

Update each parameter

and updates them using the gradients.



Training Loop
Compute gradients
Update each parameter



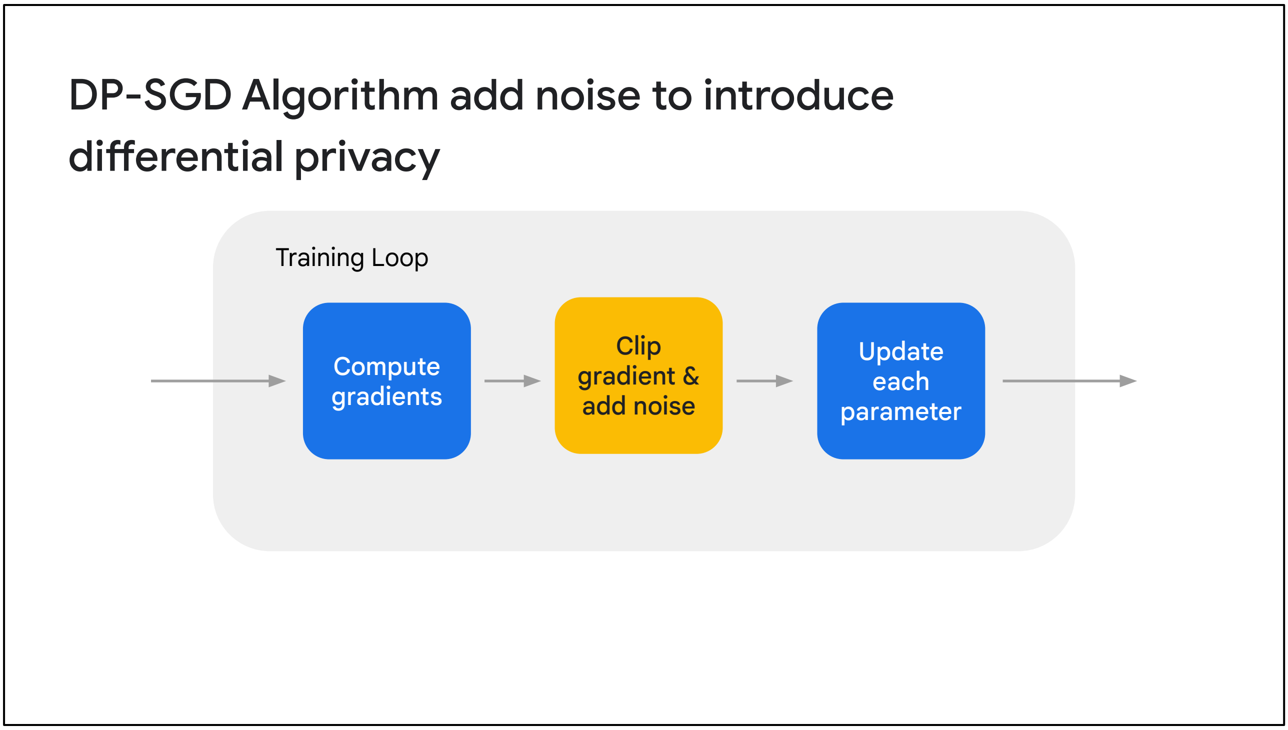
DP-SGD Algorithm add noise to introduce differential privacy

Training Loop

Compute gradients

Update each parameter

DP-SGD adds an extra step to introduce differential privacy via gradient clipping and noise addition. The amount of noise can be controlled by the sampling distribution. Larger noise is more secure, but can worsen the model performance.



DP-SGD Algorithm add noise to introduce differential privacy

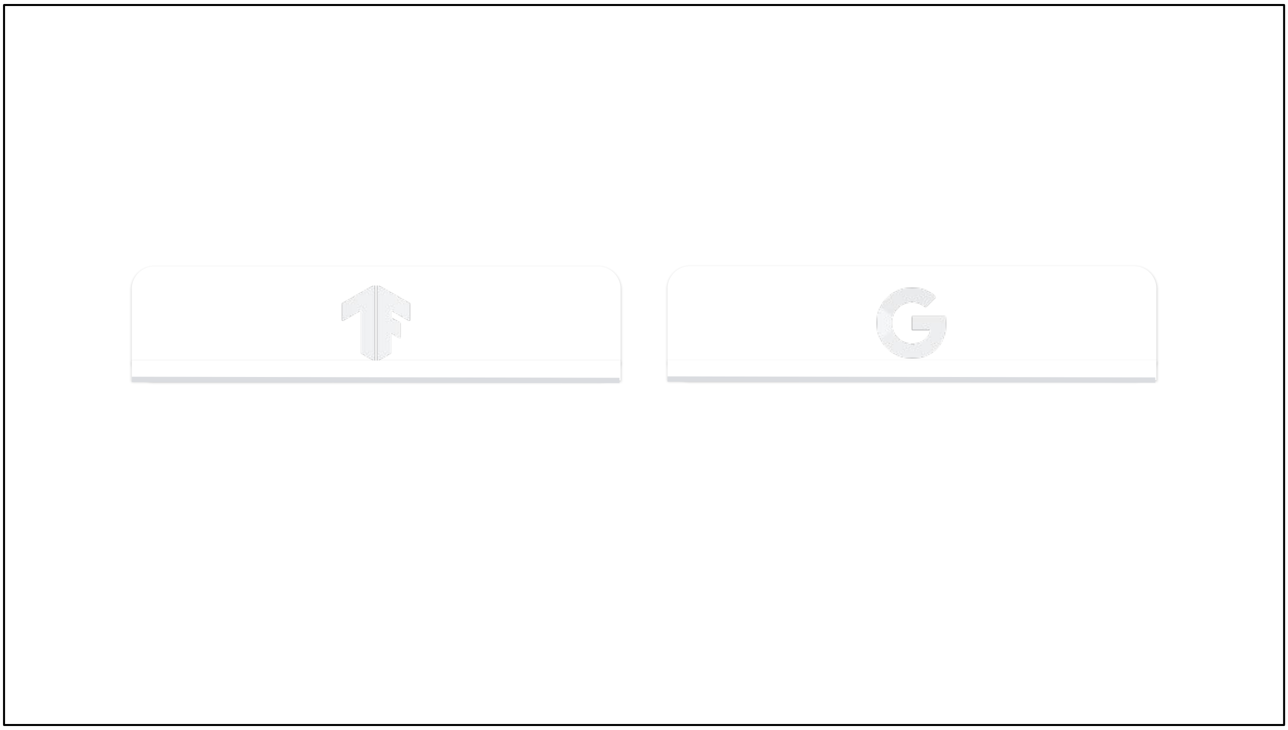
Training Loop

Compute gradients

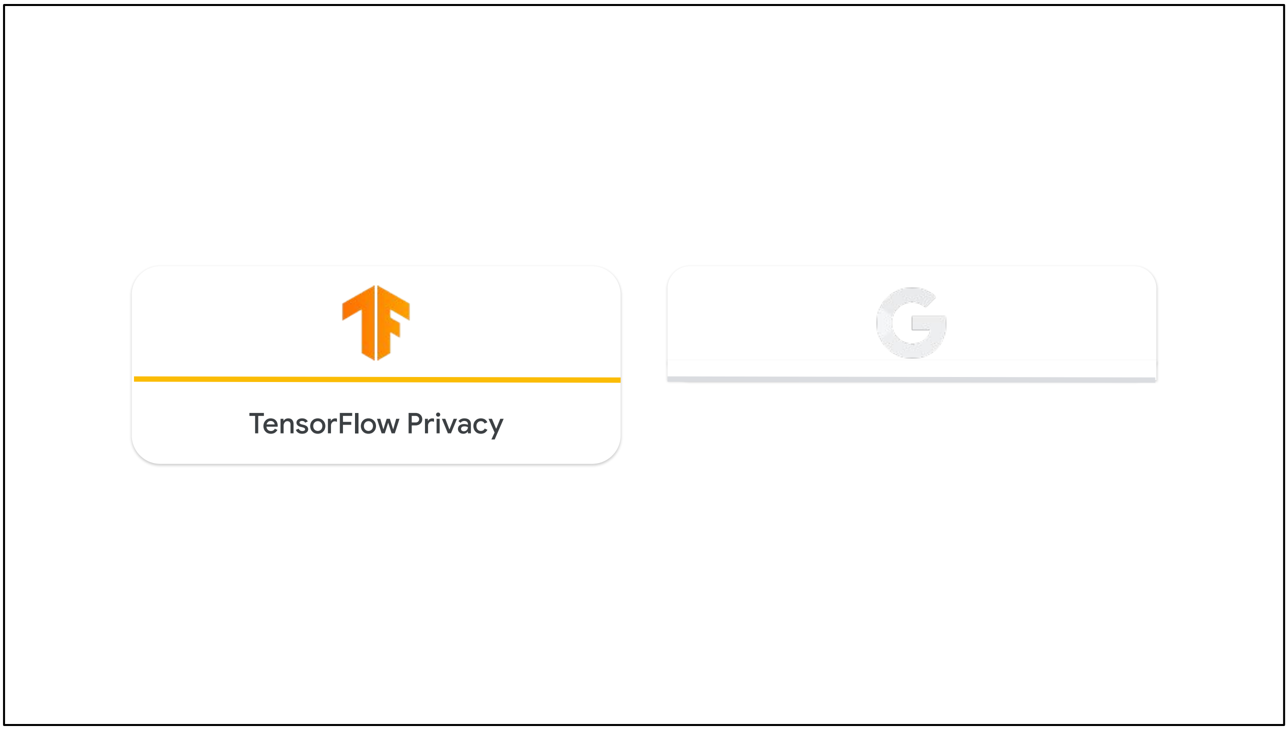
Clip gradient & add noise

Update each parameter

With this additional step, we can apply optimization in training in a more secure way.

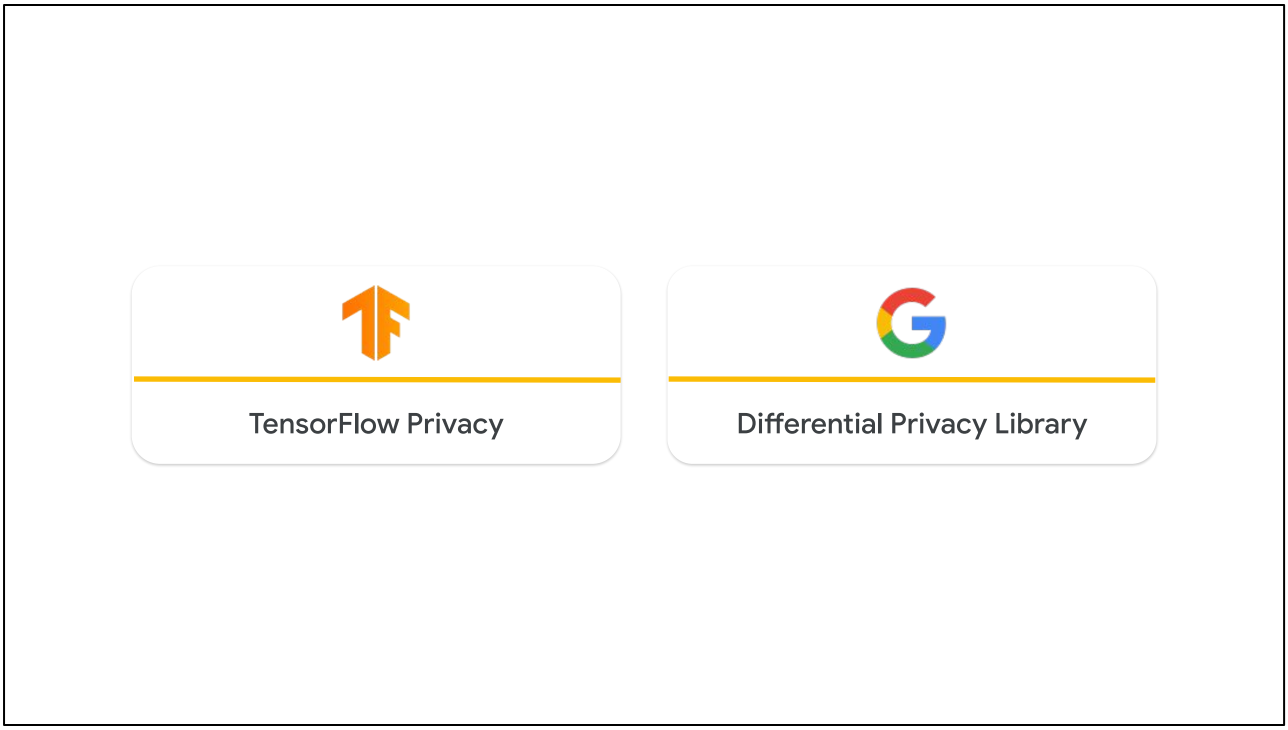


There are many improvements and additional considerations for differential privacy on machine learning training, but you can easily adopt them by using existing implementations.



TensorFlow Privacy

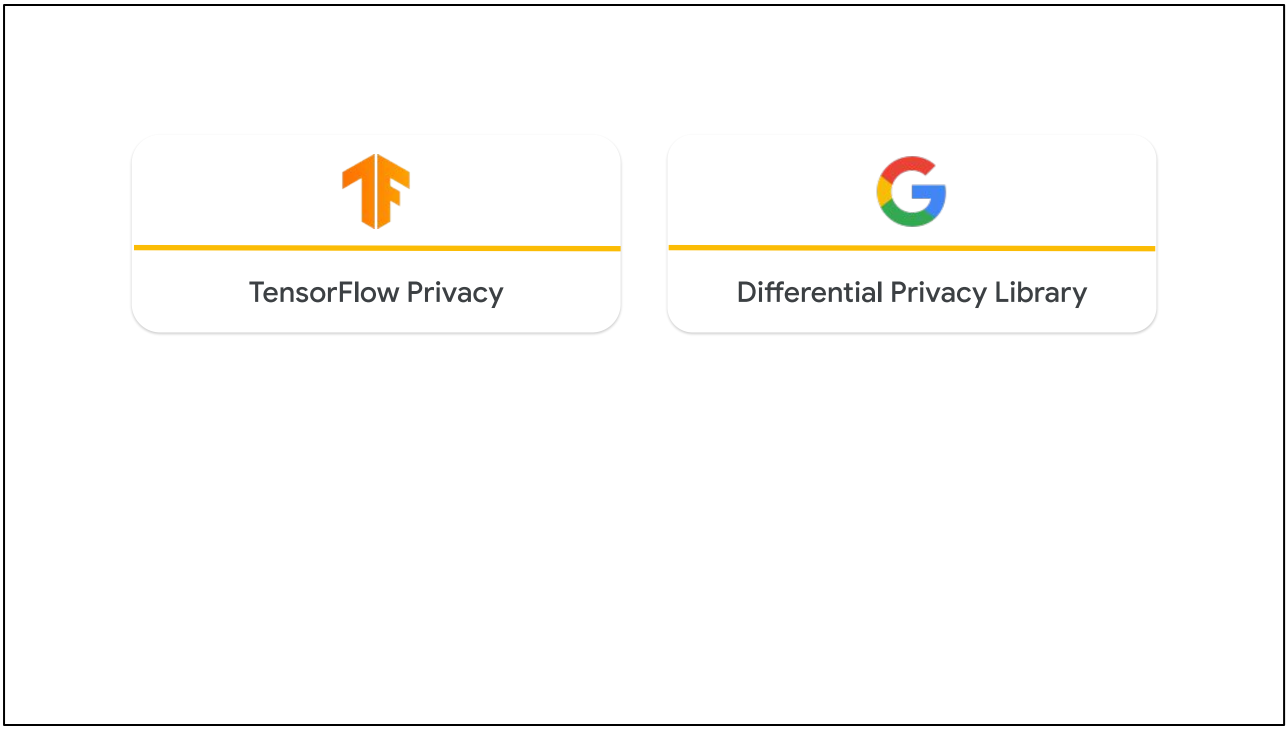
These include TensorFlow Privacy



TensorFlow Privacy

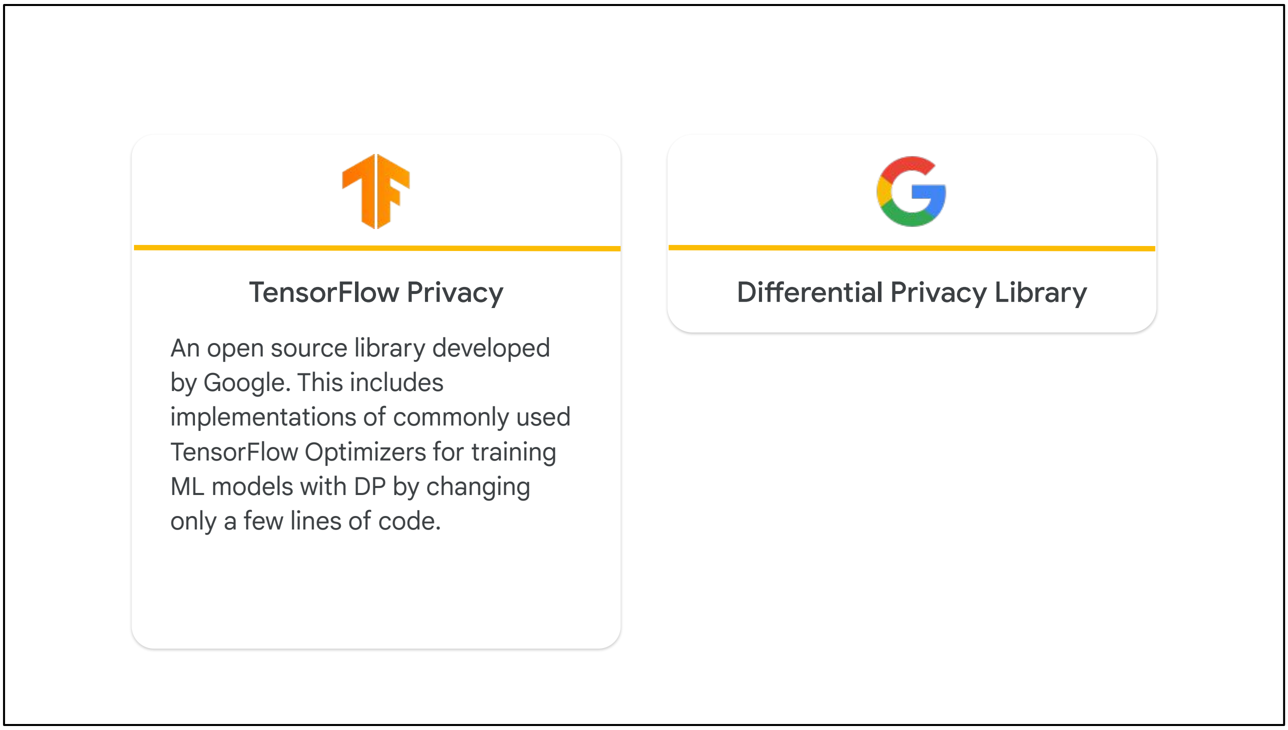
Differential Privacy Library

and Google’s Differential Privacy library.



TensorFlow Privacy

Differential Privacy Library

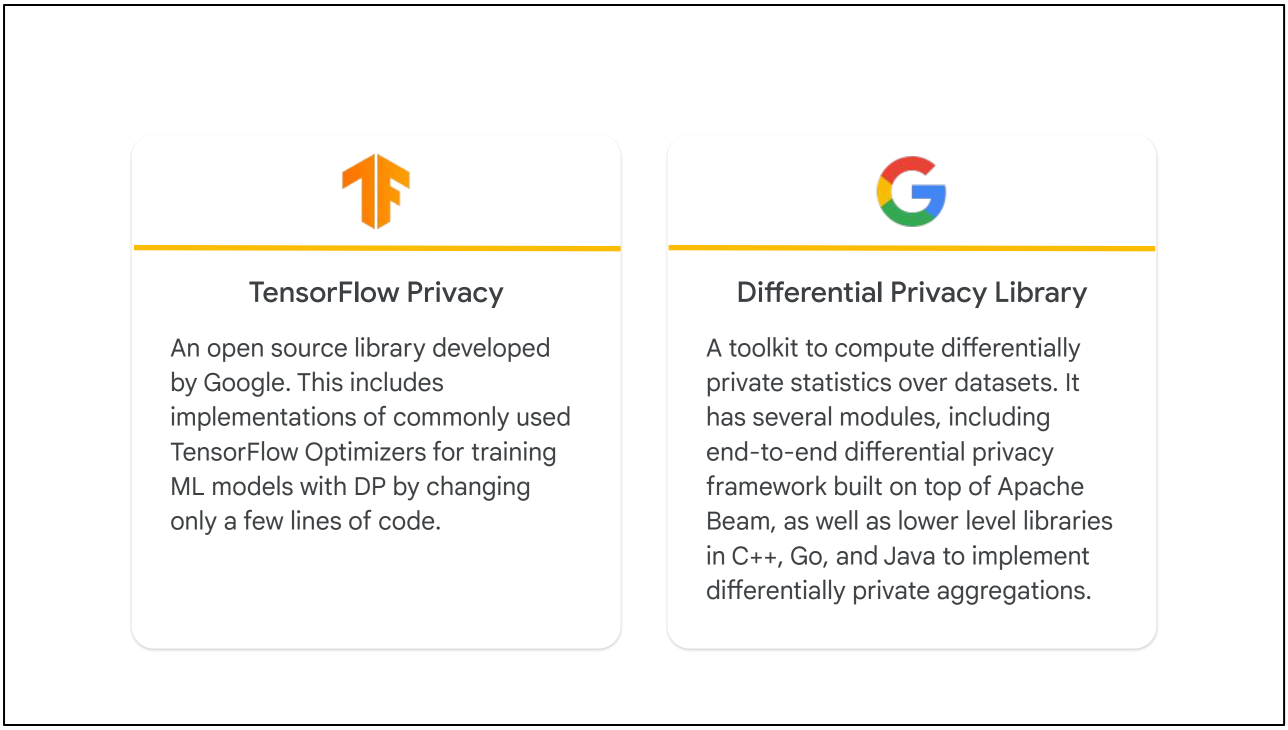


TensorFlow Privacy

An open source library developed by Google. This includes implementations of commonly used TensorFlow Optimizers for training ML models with DP by changing only a few lines of code.

Differential Privacy Library

TensorFlow Privacy is a library integrated well with the TensorFlow ecosystem. You can easily apply differentially private optimizers, including DP-SGD, to TensorFlow models using this library.



TensorFlow Privacy

An open source library developed by Google. This includes implementations of commonly used TensorFlow Optimizers for training ML models with DP by changing only a few lines of code.

Differential Privacy Library

A toolkit to compute differentially private statistics over datasets. It has several modules, including end-to-end differential privacy framework built on top of Apache Beam, as well as lower level libraries in C++, Go, and Java to implement differentially private aggregations.

Google’s Differential Privacy library is a more generic library for differential privacy, as it is used not only for machine learning purposes. It includes an end-to-end differential privacy framework built on top of Apache Beam, as well as lower level libraries in C++, Go, and Java to implement differentially private aggregations.

01 Overview of AI Privacy

02 Privacy in Training Data: De-identification techniques

03 Privacy in Training Data: Randomization techniques

04 Privacy in Machine Learning Training: DP-SGD

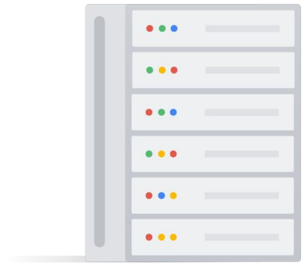
05 Privacy in Machine Learning Training: Federated Learning

06 System Security on Google Cloud

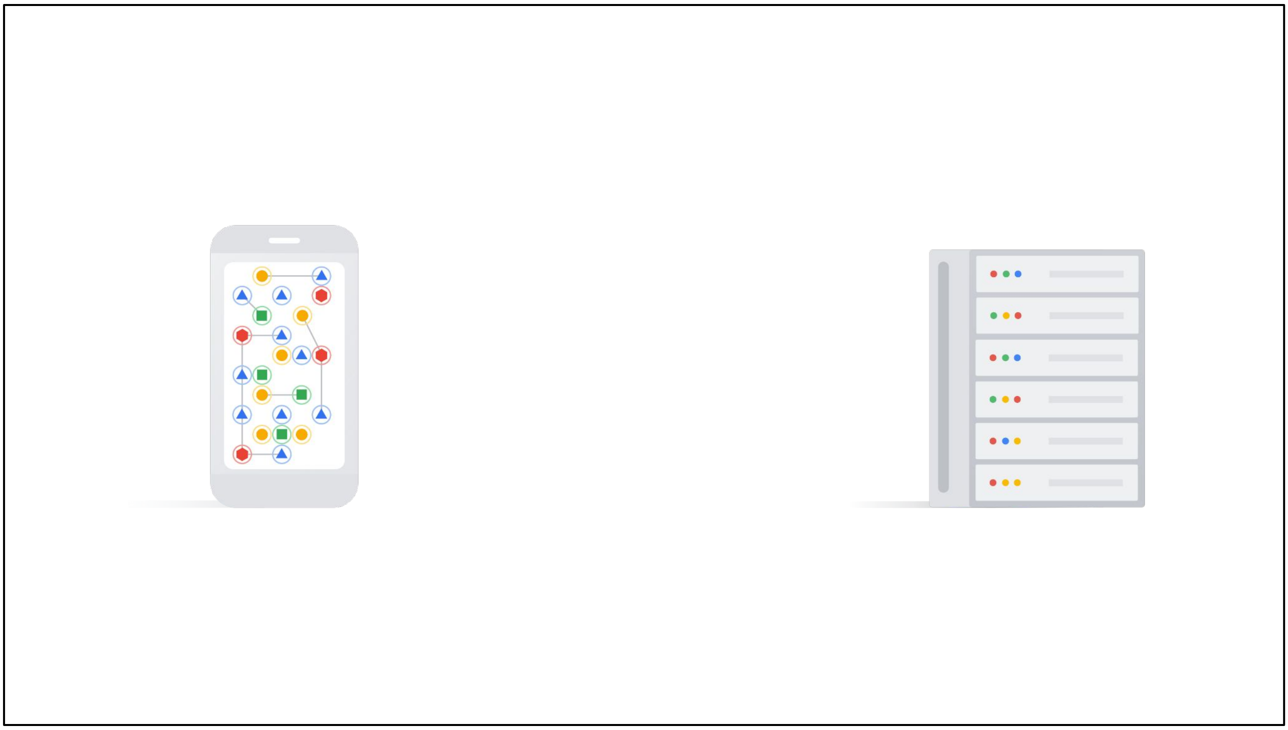
07 System Security and Generative AI

08 Lab: Differential Privacy in Machine Learning with TensorFlow Privacy

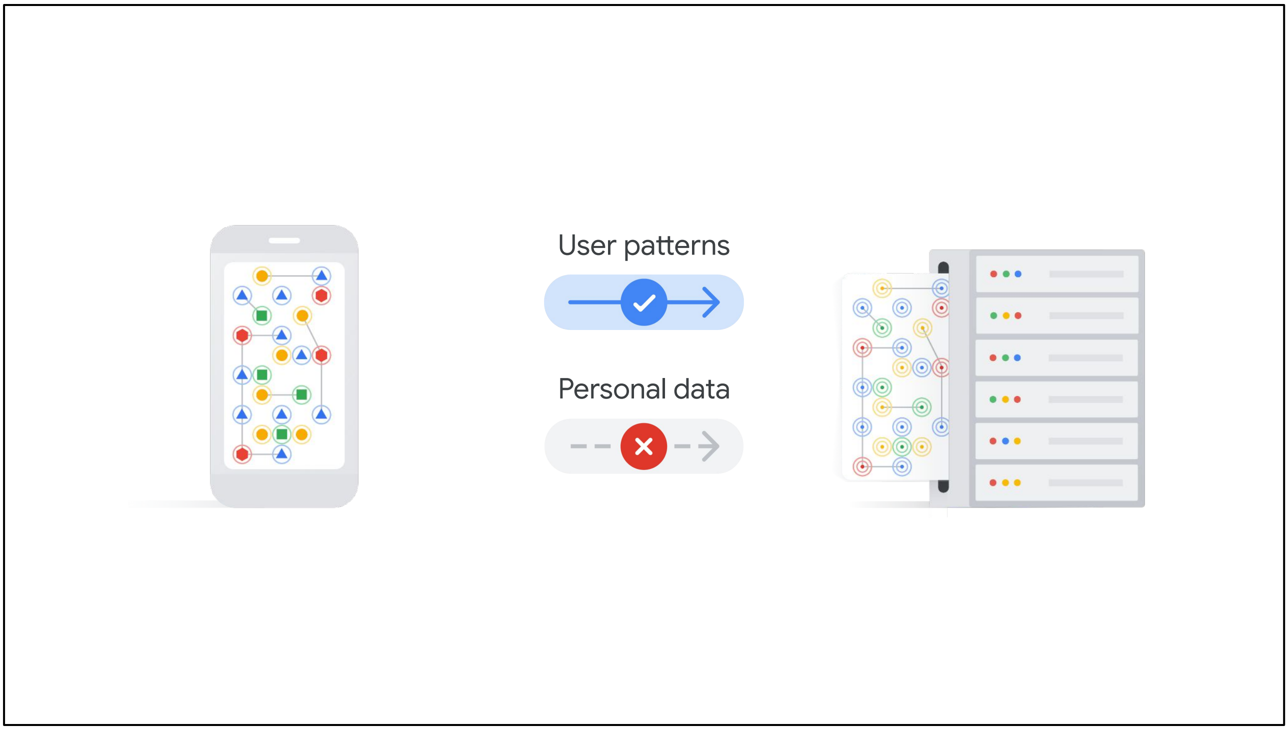
Now, let’s move on to the next methodology, federated learning.



Federated learning is a useful method for training large-scale machine learning models



using edge devices (such as smartphones)

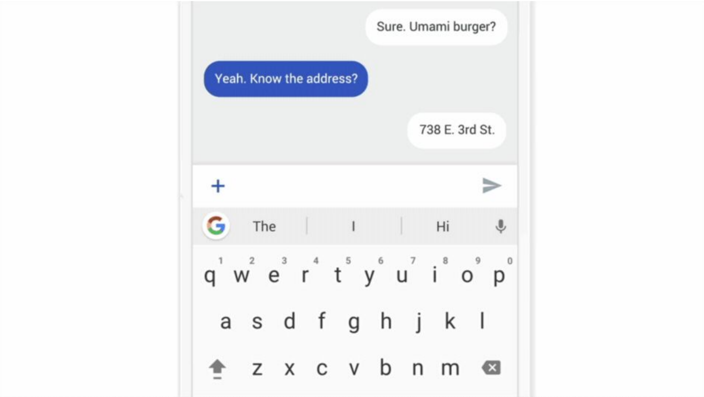


User patterns

Personal data

while preserving data privacy by avoiding the need to share raw data.

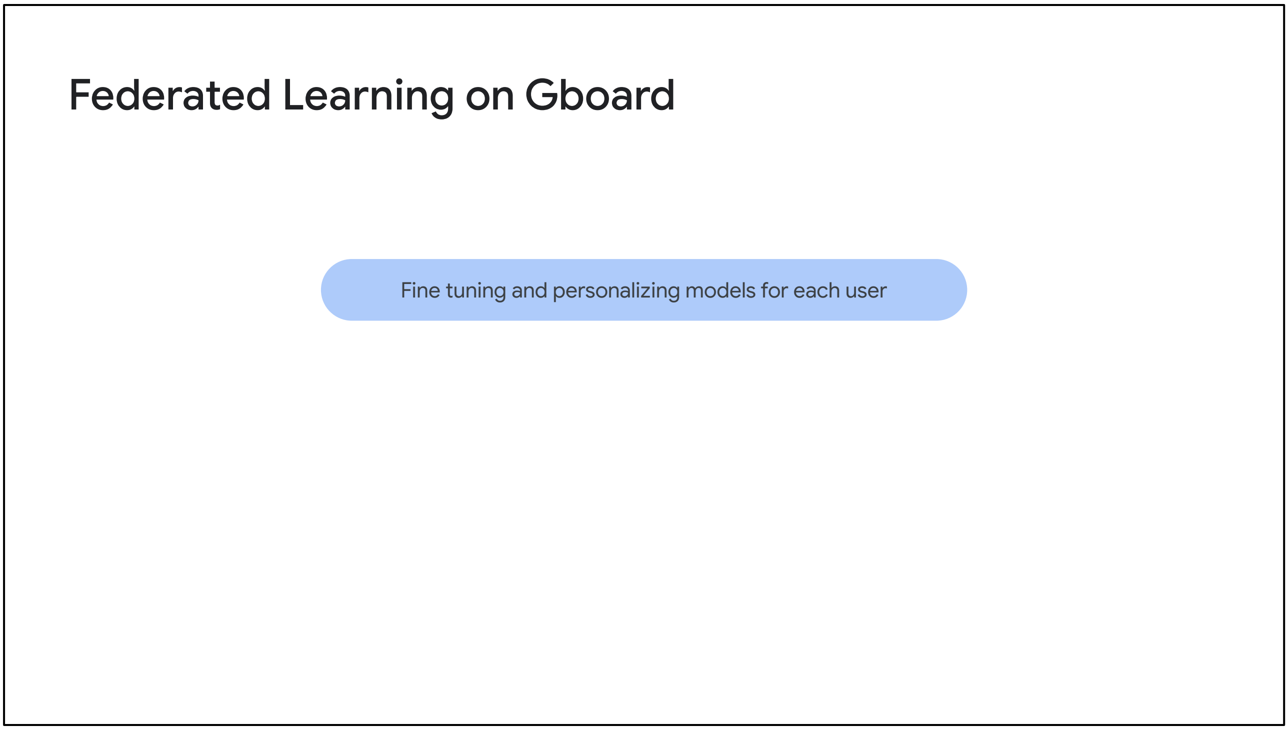
Federated Learning on Gboard


Sure. Umami burger?

Yeah. Know the address?

738 E. 3rd St.

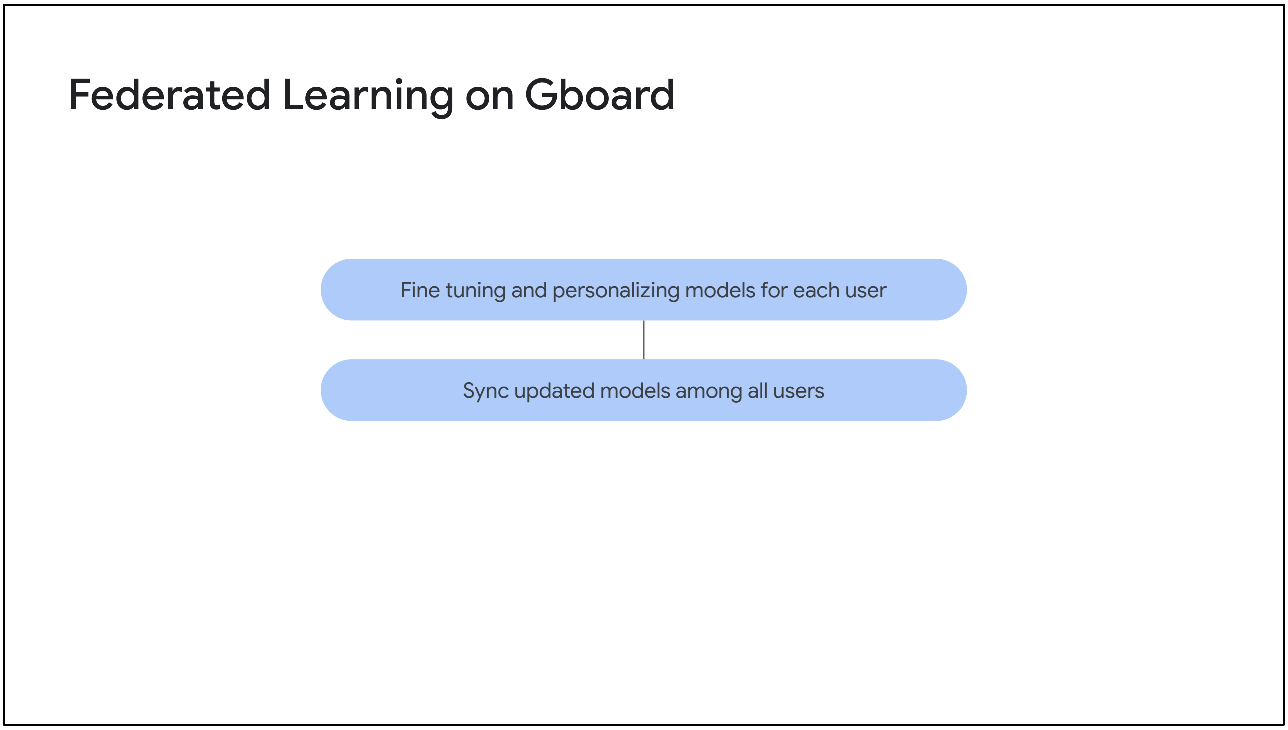
This technique is adopted in many Google services and applications, including Gboard.
Gboard has several machine learning models to enhance users' typing experiences on smartphones,



Federated Learning on Gboard

Fine tuning and personalizing models for each user

based on users’ inputs.
Since every user has different usages and needs, fine-tuning and personalizing the models for each user is critical to support unique needs,

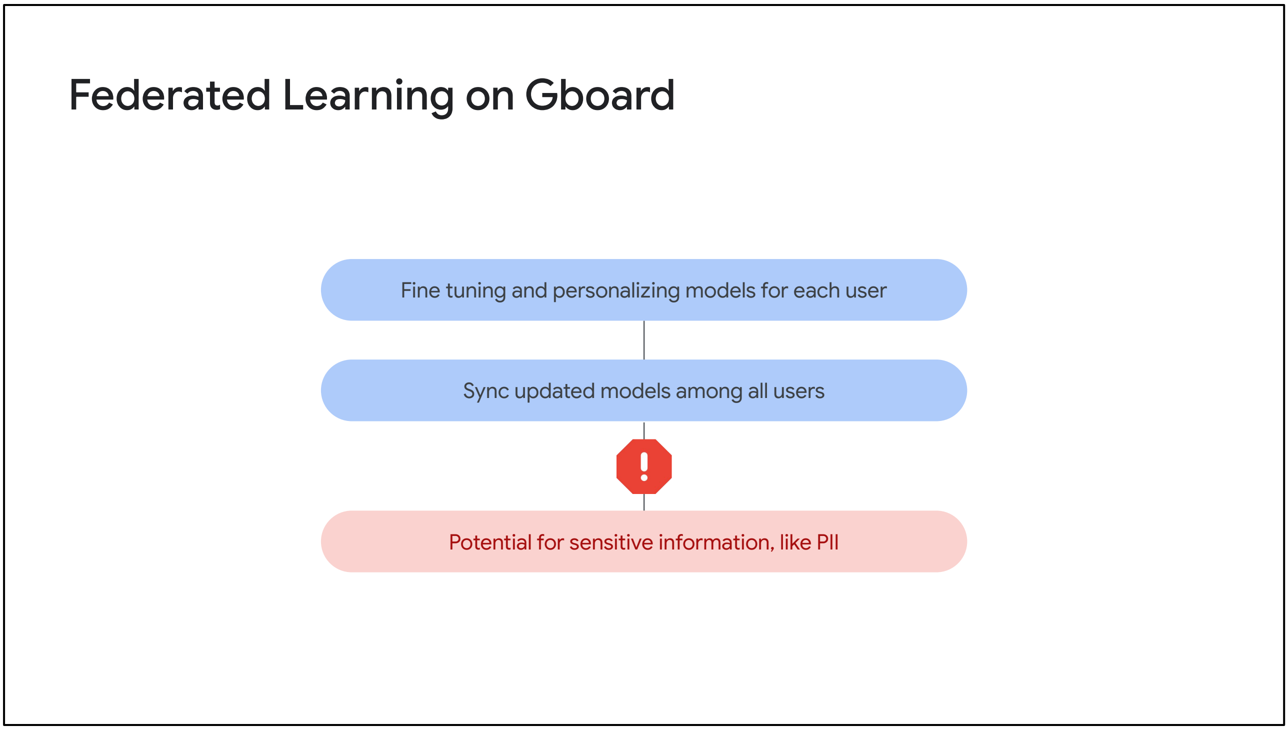


Federated Learning on Gboard

Fine tuning and personalizing models for each user

Sync updated models among all users

while it is also important to sync updated models among all users to improve the product itself.



Federated Learning on Gboard

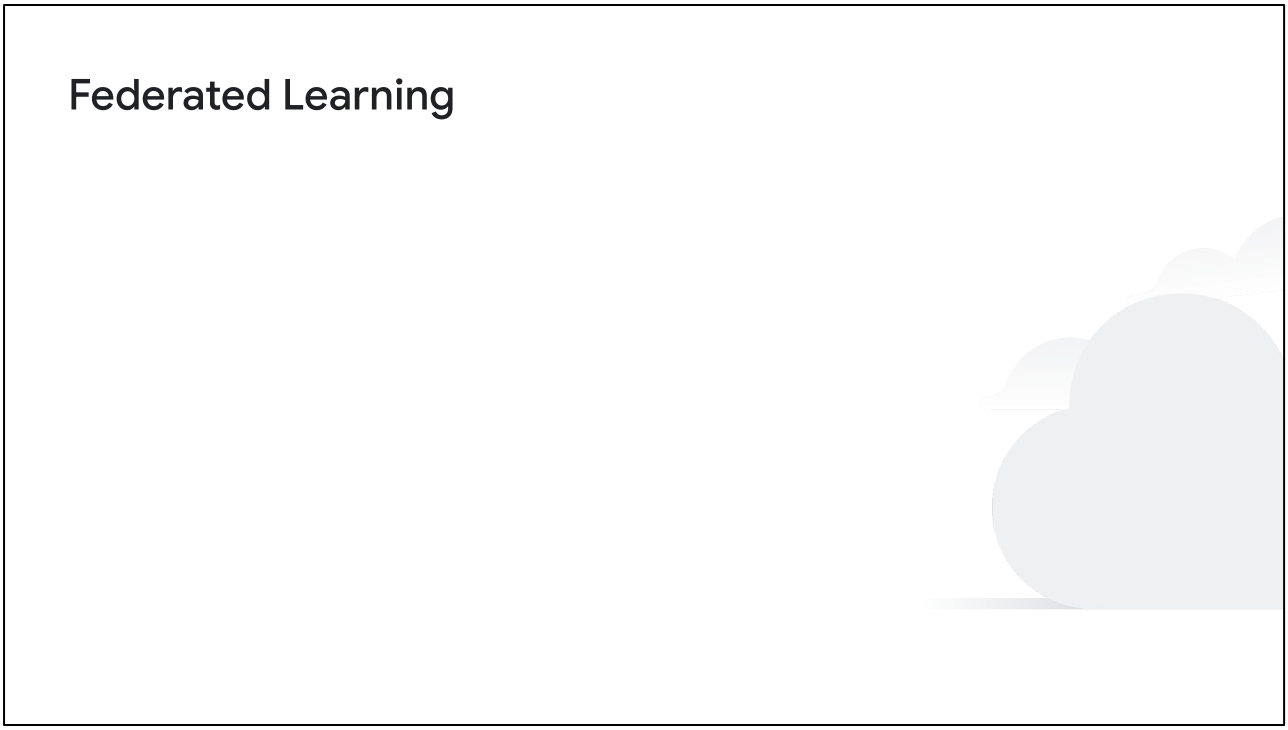
Fine tuning and personalizing models for each user

Sync updated models among all users

Potential for sensitive information, like PII

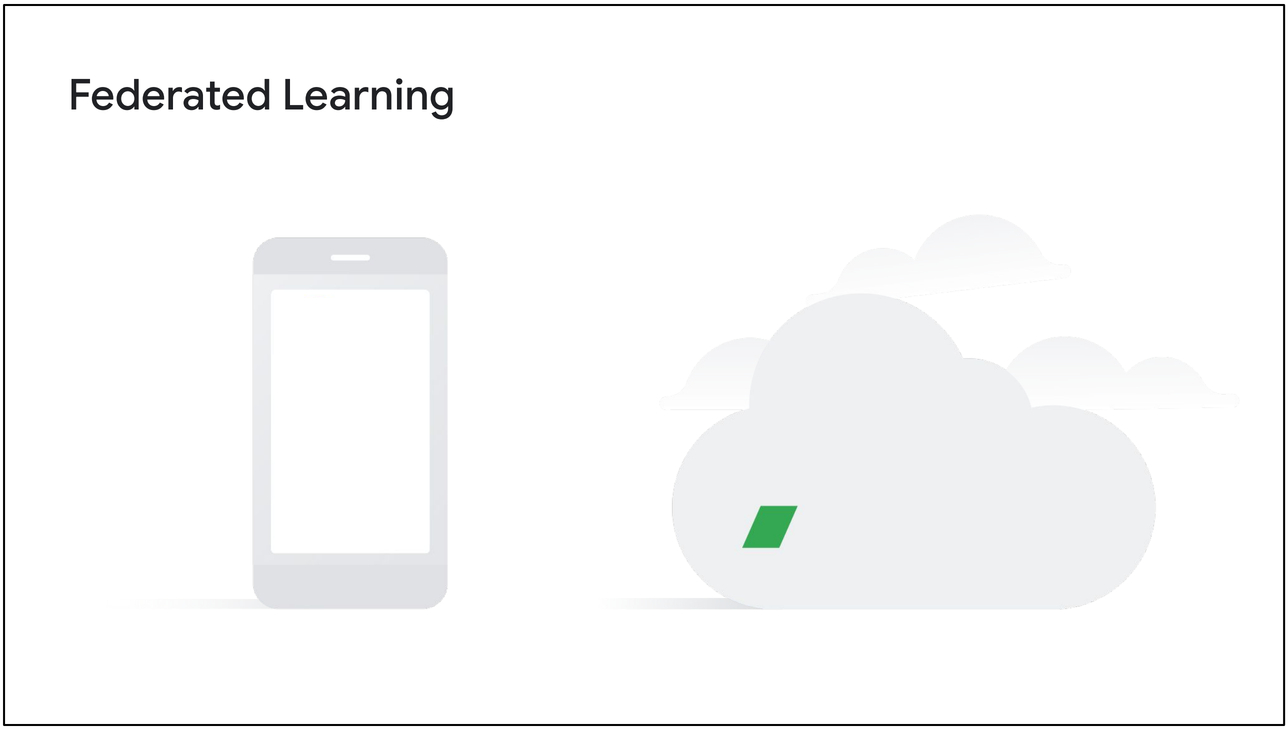
However, the typing input can contain sensitive information including PII, which users don’t want to send out to a central server for machine learning training.

In this use case, federated learning is a critical method for ensuring both privacy and personalization. Google has been implementing many systems that use this technology.



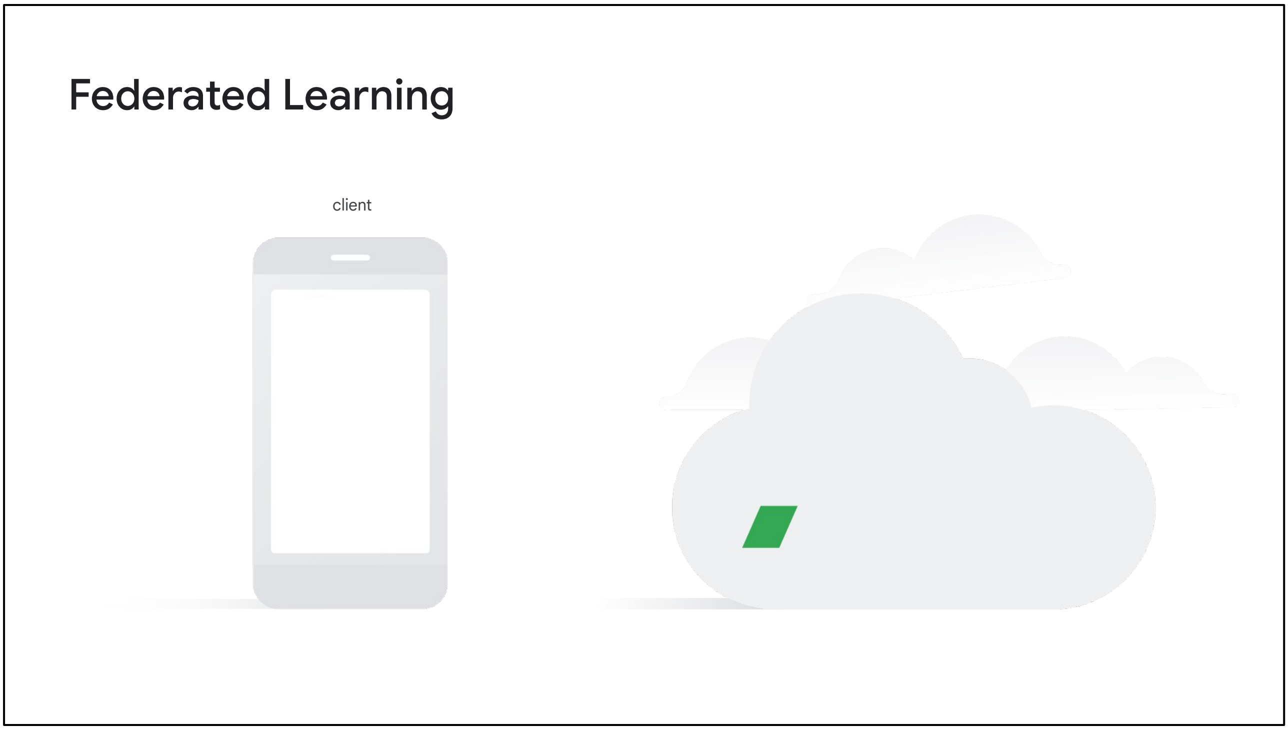
Federated Learning

Let’s take a look at how it works.



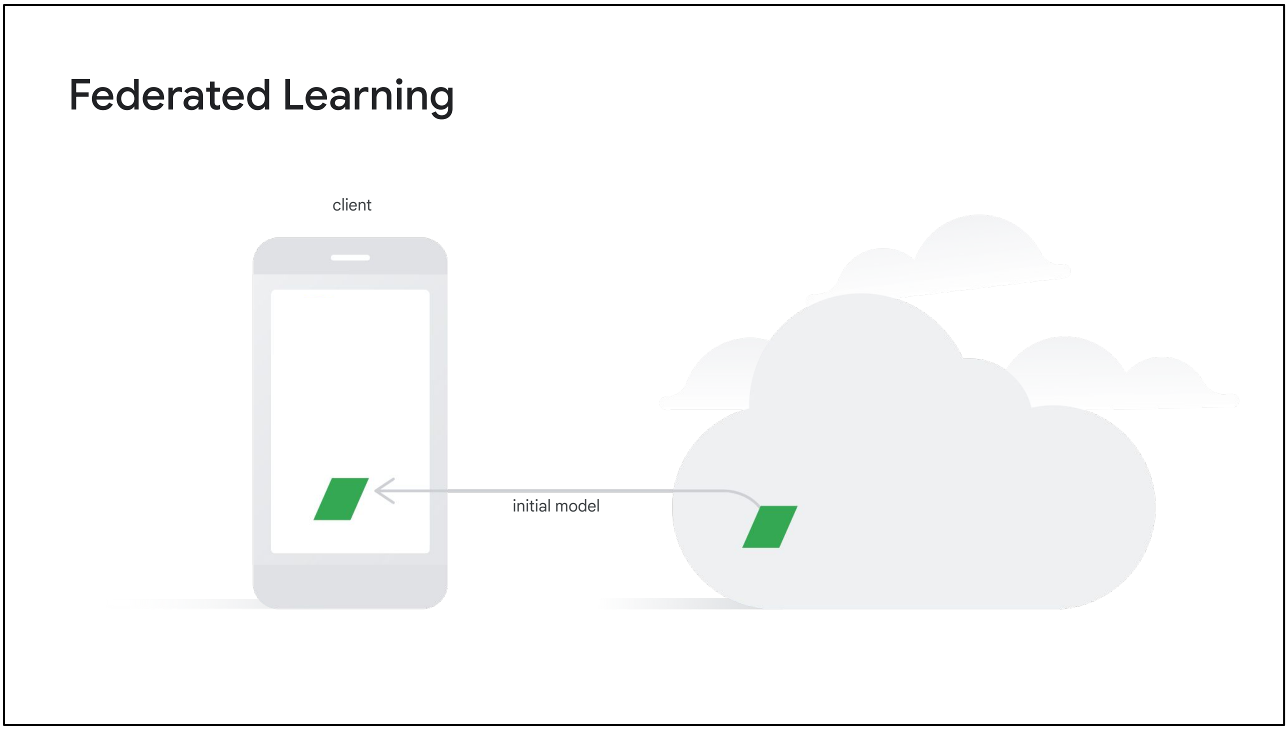
Federated Learning

Imagine you built a mobile application which includes a machine learning model.



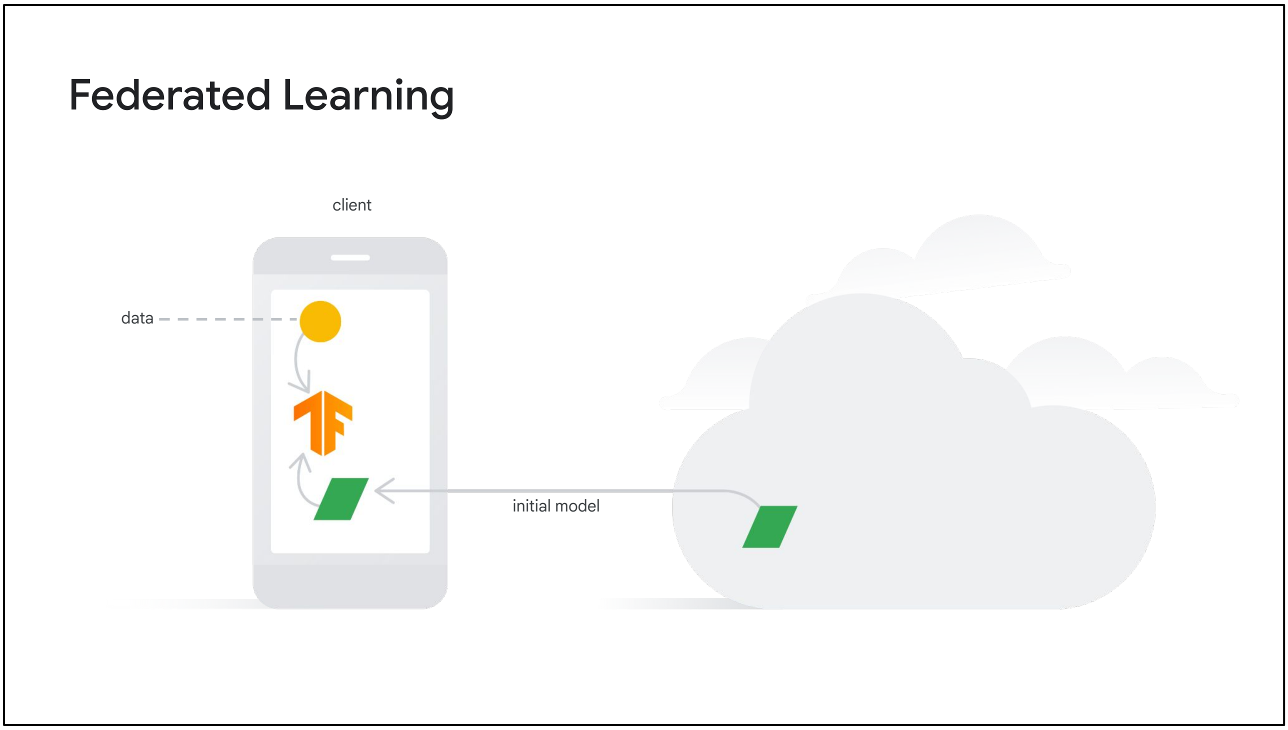
Federated Learning

You trained the first model with a standard dataset,



Federated Learning

and distributed the app to users' devices.



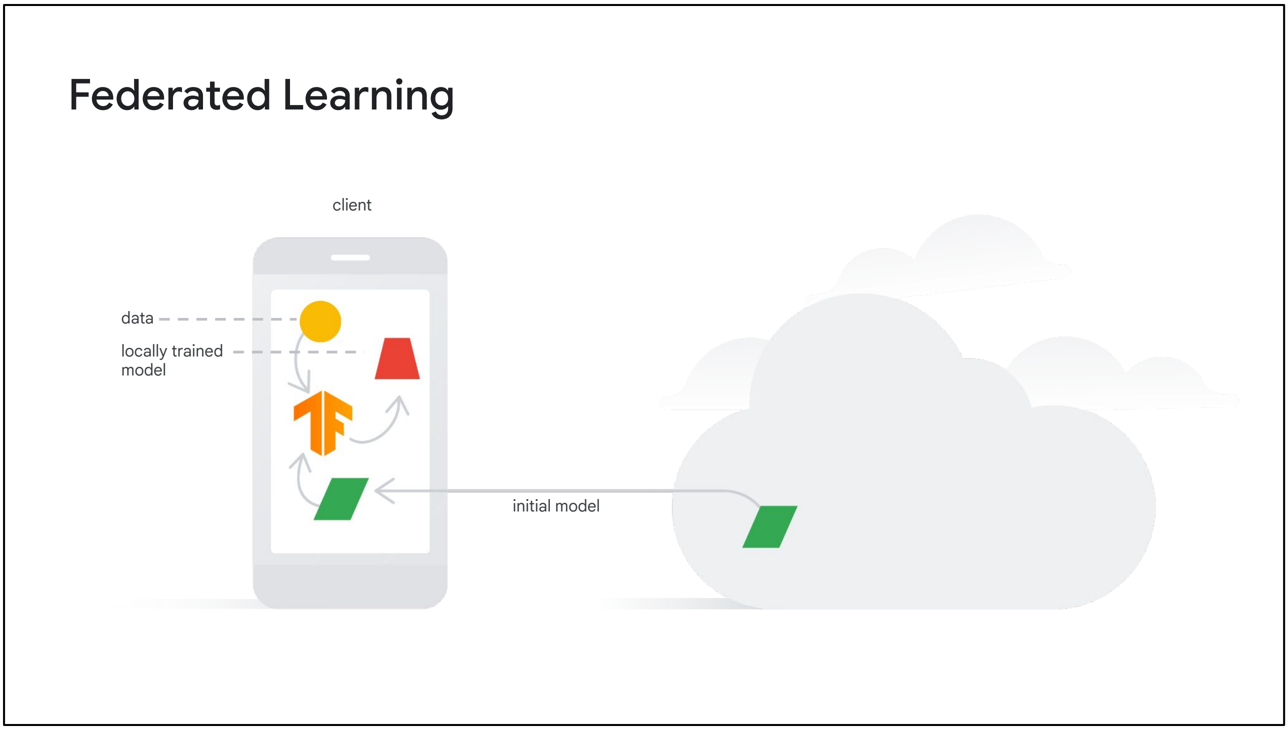
Federated Learning

client

data

initial model

Similar to how Gboard personalization works, Federated Learning can



Federated Learning

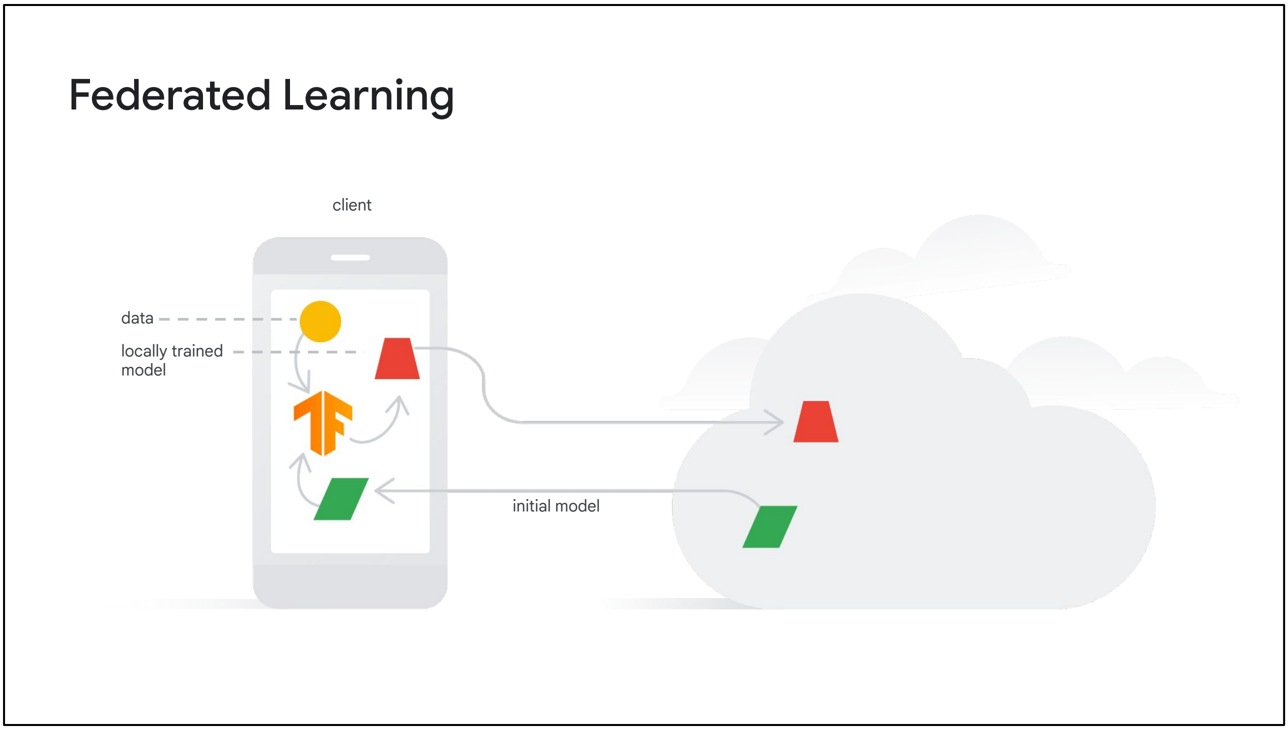
client

data

locally trained model

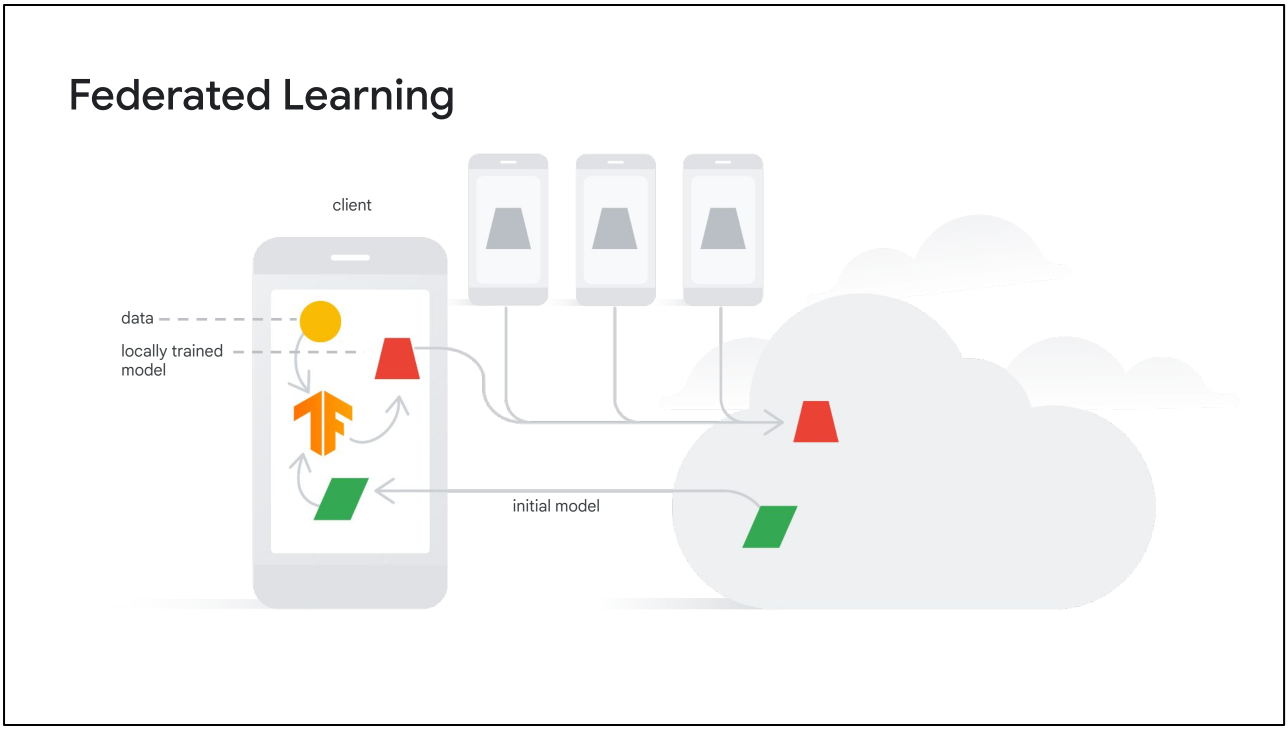
initial model

update machine learning models on user devices (e.g. smartphones) using their own data.



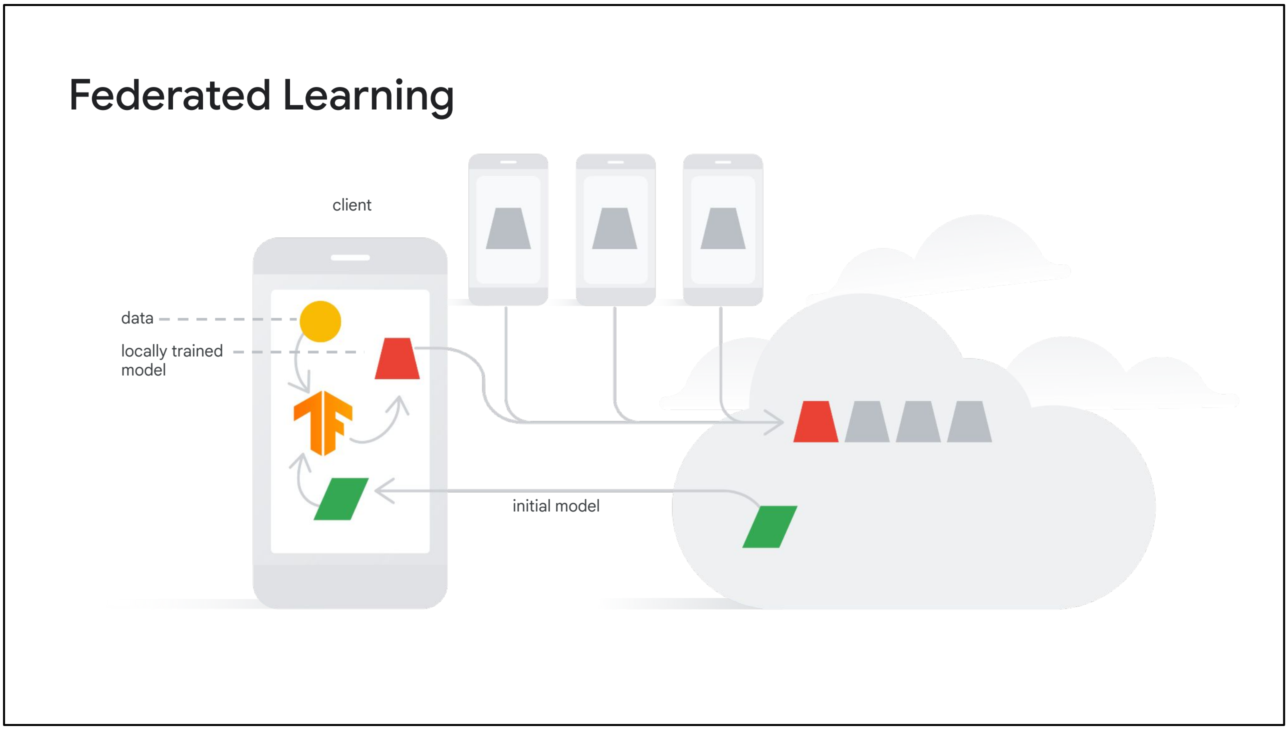
Federated Learning

After these local updates, the improved model parameters (gradients) are shared with the central server so that the changes are reflected in the central model. This way, the central model is enhanced with updated models using fresh data that benefits all users without compromising their privacy.



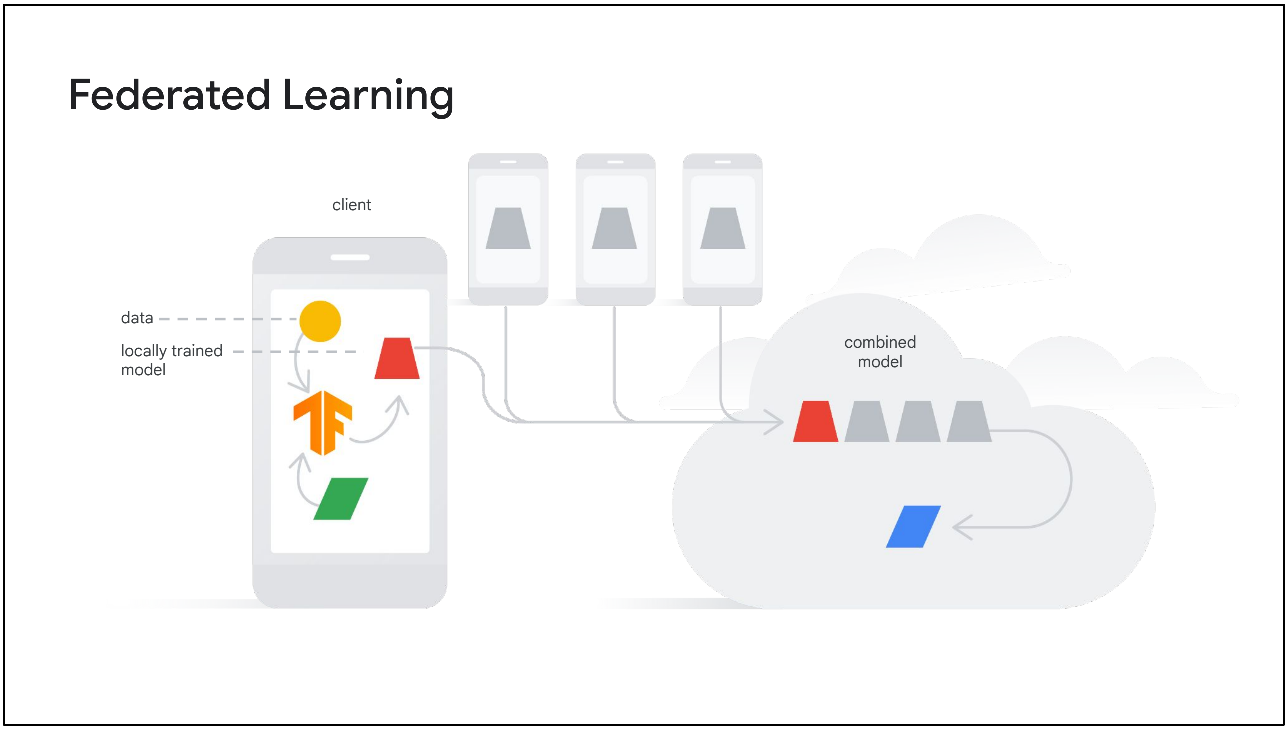
Federated Learning

Lots of update information is gathered from the numerous user devices without collecting any raw data from those devices.



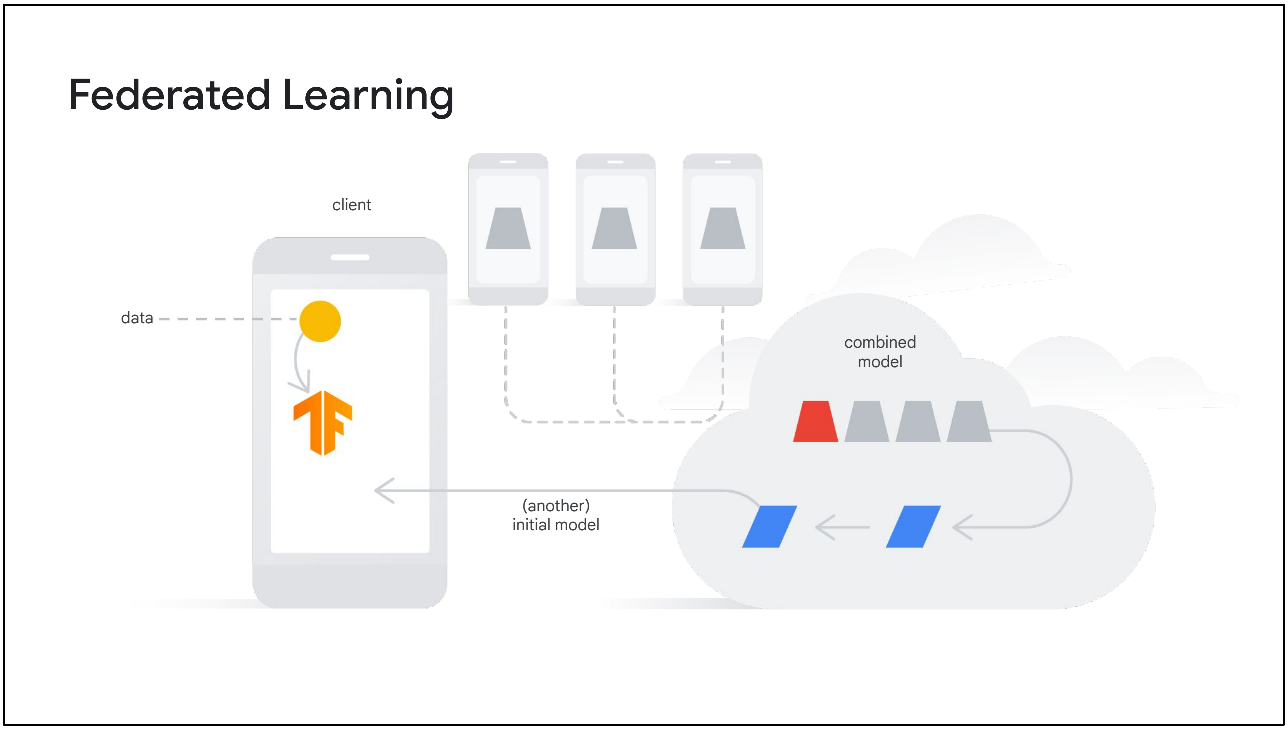
Federated Learning

Practically, you should carefully design the user sampling method to avoid unintended results like data skew.



Federated Learning

Then, aggregate the gradients to update the central model.



Federated Learning

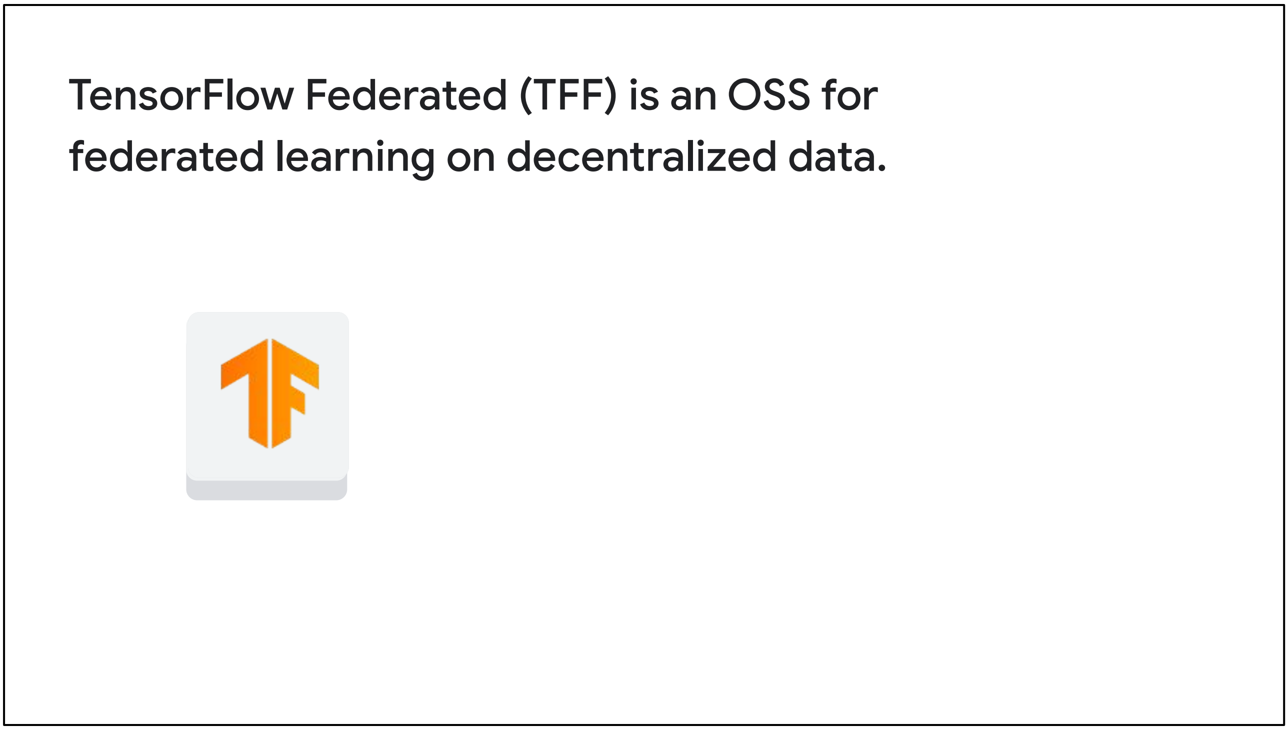
client

data

combined
model

(another)
initial model

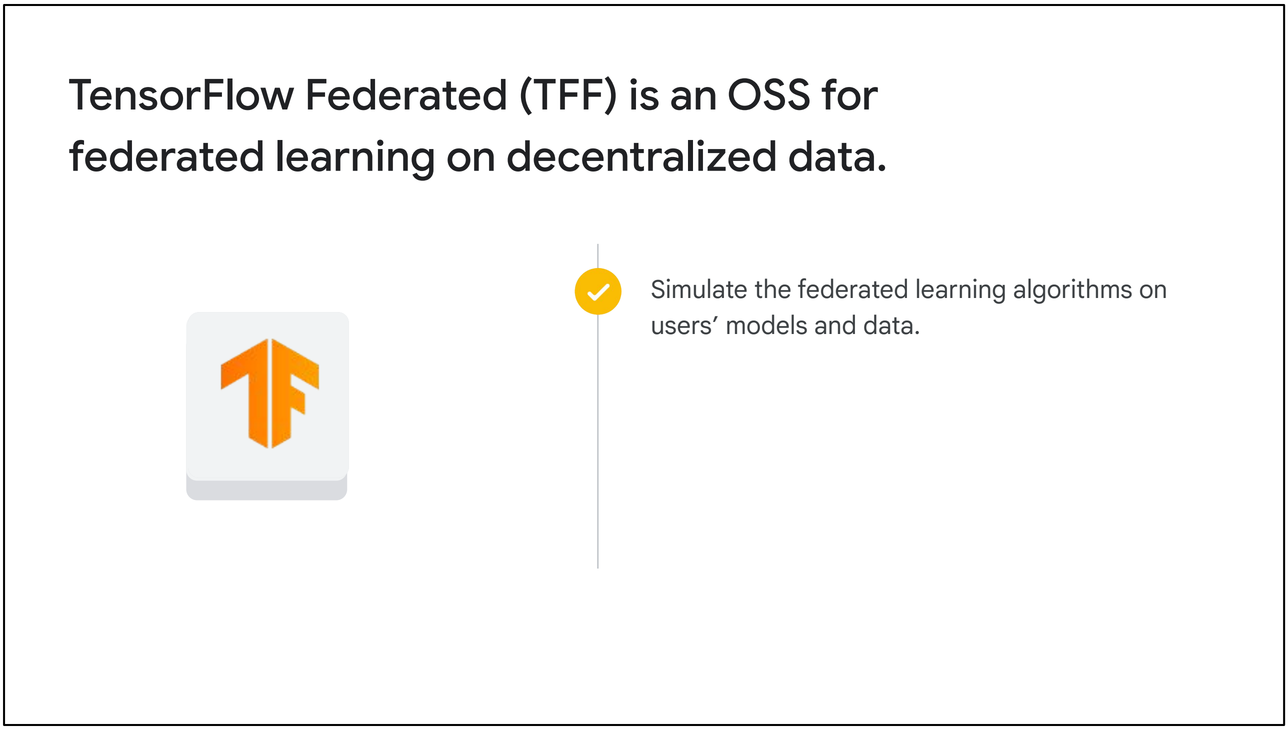
By following these federated learning steps, you can update models using actual users' data, while keeping them on each device.
This can be repeated to update the model.



TensorFlow Federated (TFF) is an OSS for federated learning on decentralized data.

Although the concept of federated learning may seem straightforward, its implementation and configuration is tricky since it requires multiple systems working together.

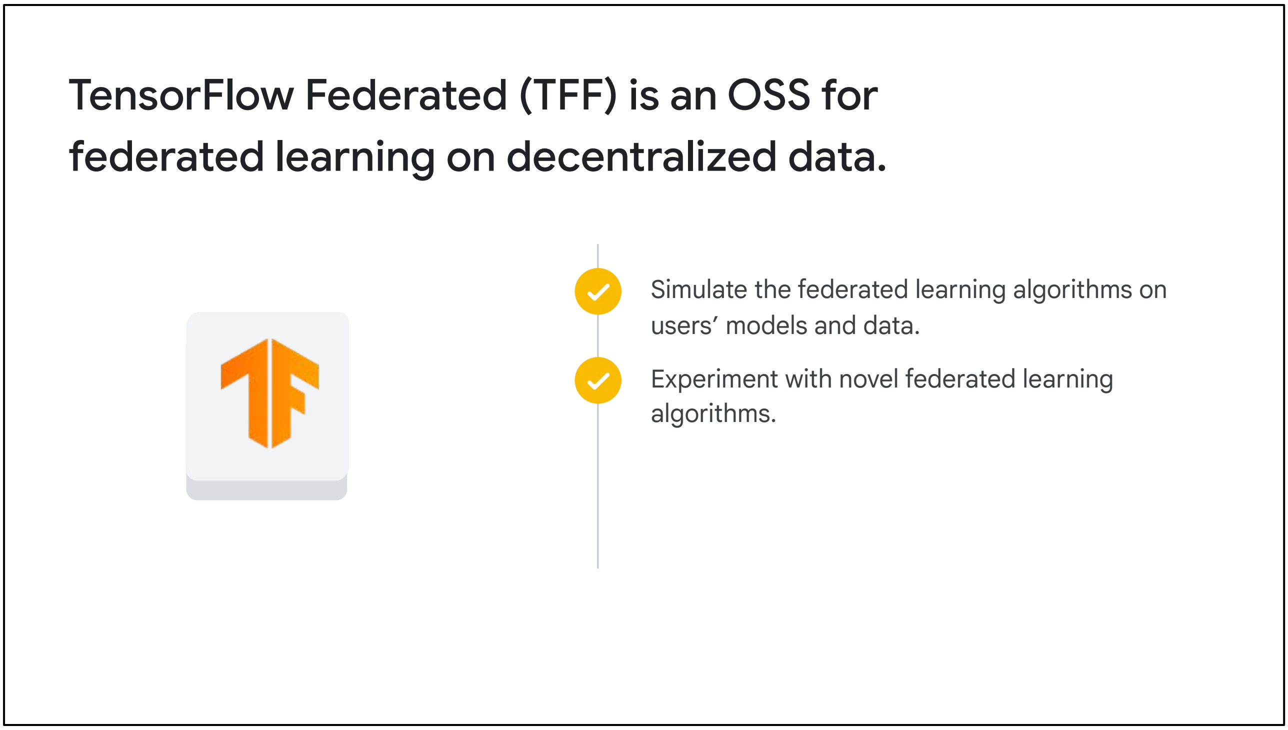
TensorFlow Federated, or TFF, offers a useful toolbox for this purpose.



TensorFlow Federated (TFF) is an OSS for federated learning on decentralized data.

✓ Simulate the federated learning algorithms on users’ models and data.

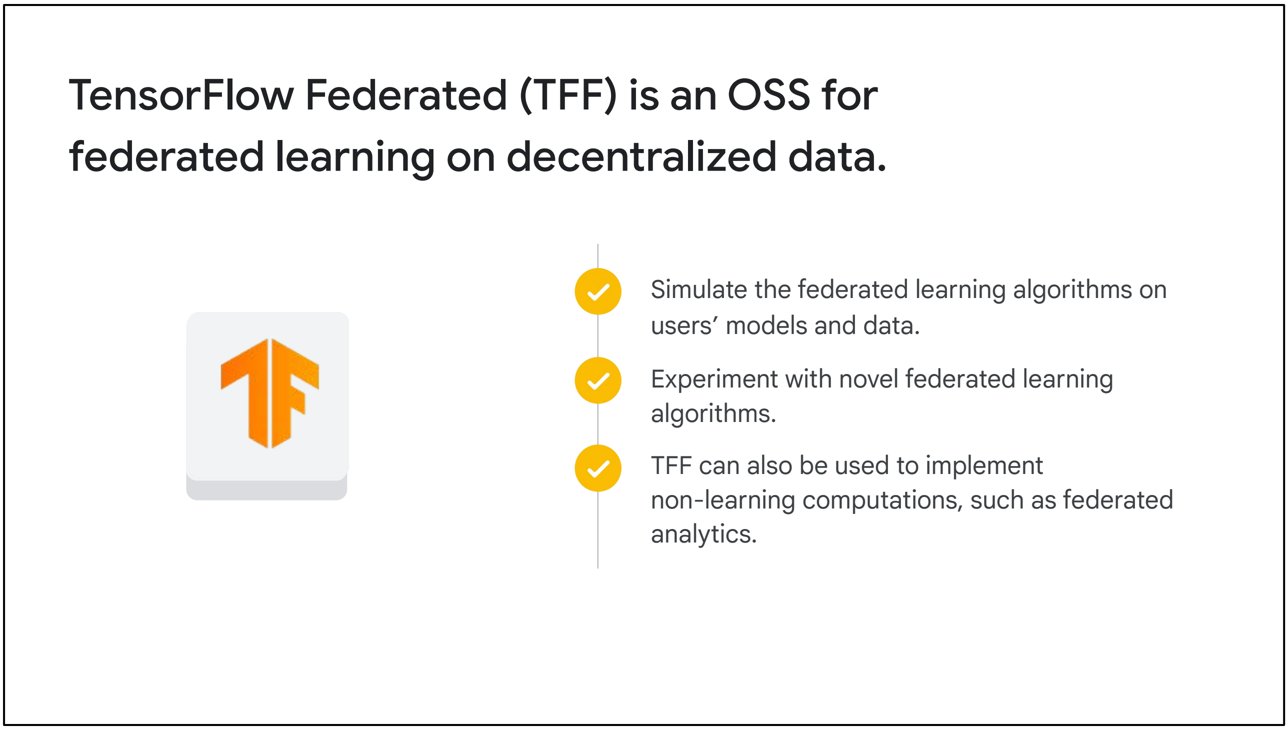
With TFF, you can simulate the federated learning environment quickly,



TensorFlow Federated (TFF) is an OSS for federated learning on decentralized data.

- ✓ Simulate the federated learning algorithms on users' models and data.
- ✓ Experiment with novel federated learning algorithms.

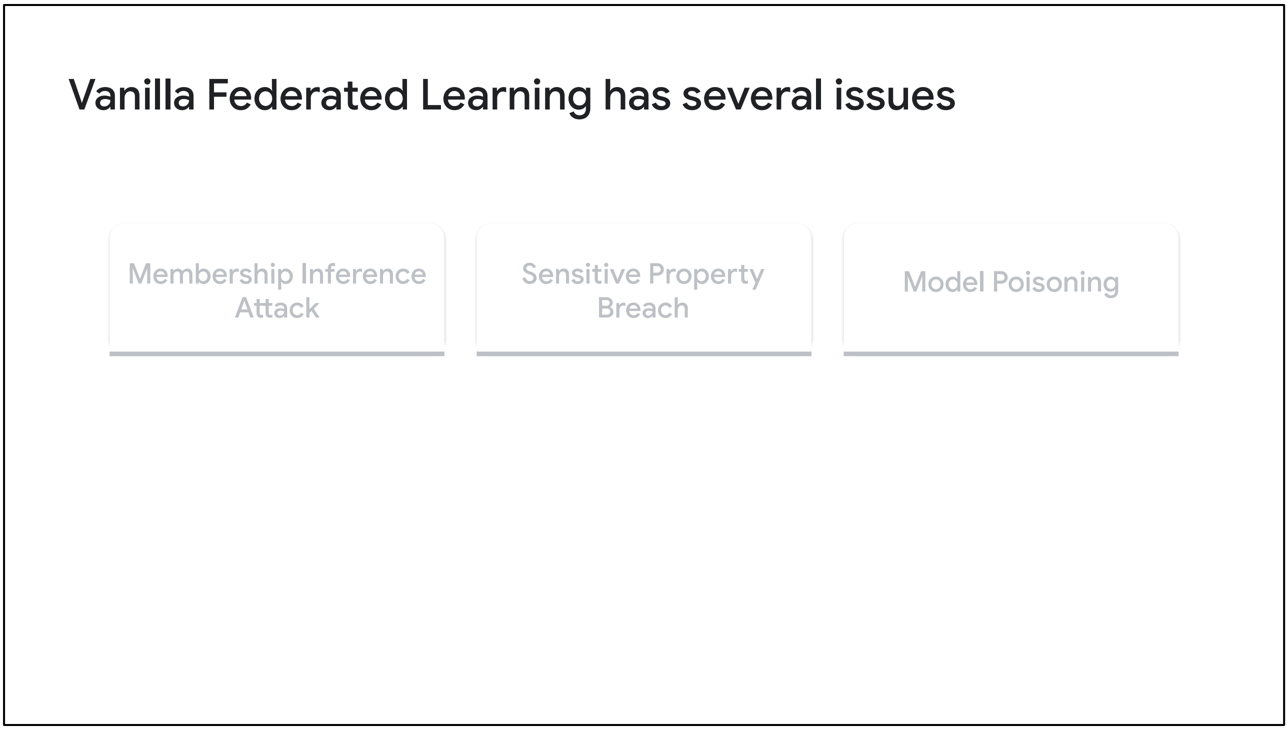
and apply complex novel algorithms.



TensorFlow Federated (TFF) is an OSS for federated learning on decentralized data.

- ✓ Simulate the federated learning algorithms on users’ models and data.
- ✓ Experiment with novel federated learning algorithms.
- ✓ TFF can also be used to implement non-learning computations, such as federated analytics.

You can even use TFF for non-machine-learning training purposes, like computing the mean of all user behavior. This is called federated analytics.



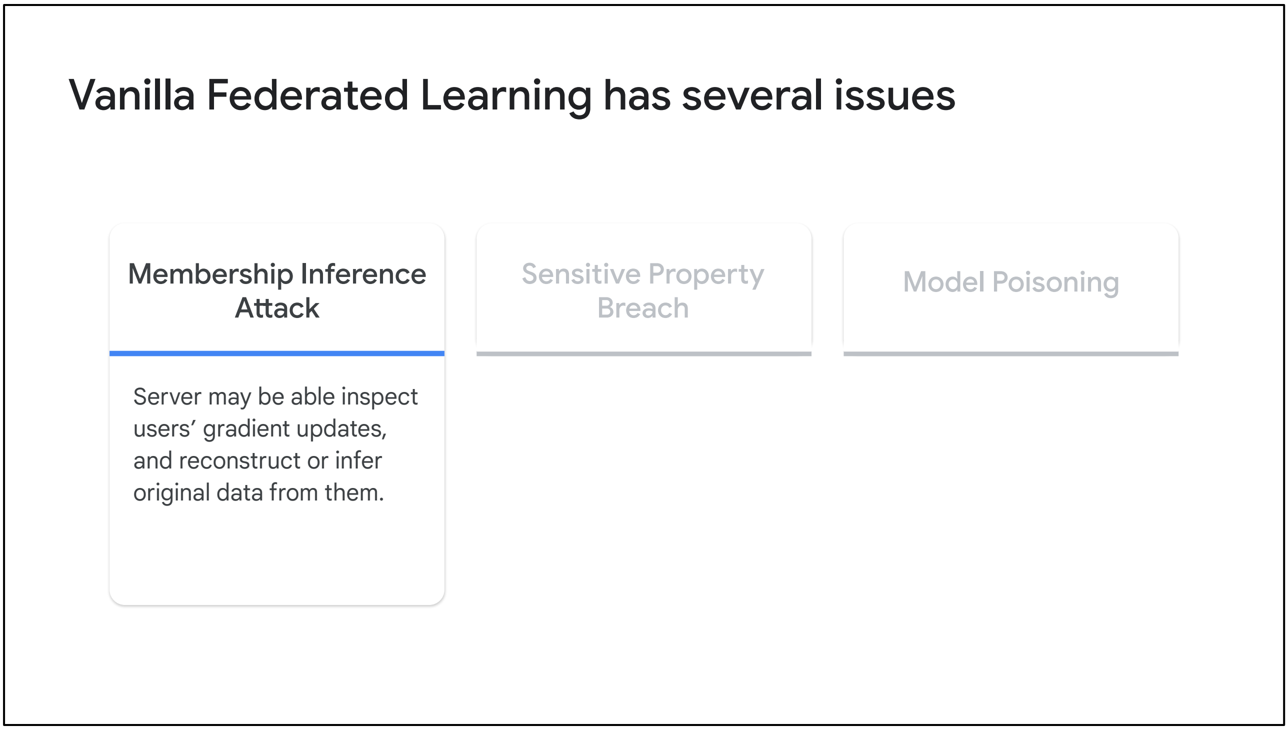
Vanilla Federated Learning has several issues

Membership Inference Attack

Sensitive Property Breach

Model Poisoning

Federated learning technology sounds cool, but it still has some issues we need to tackle. For example:



Vanilla Federated Learning has several issues

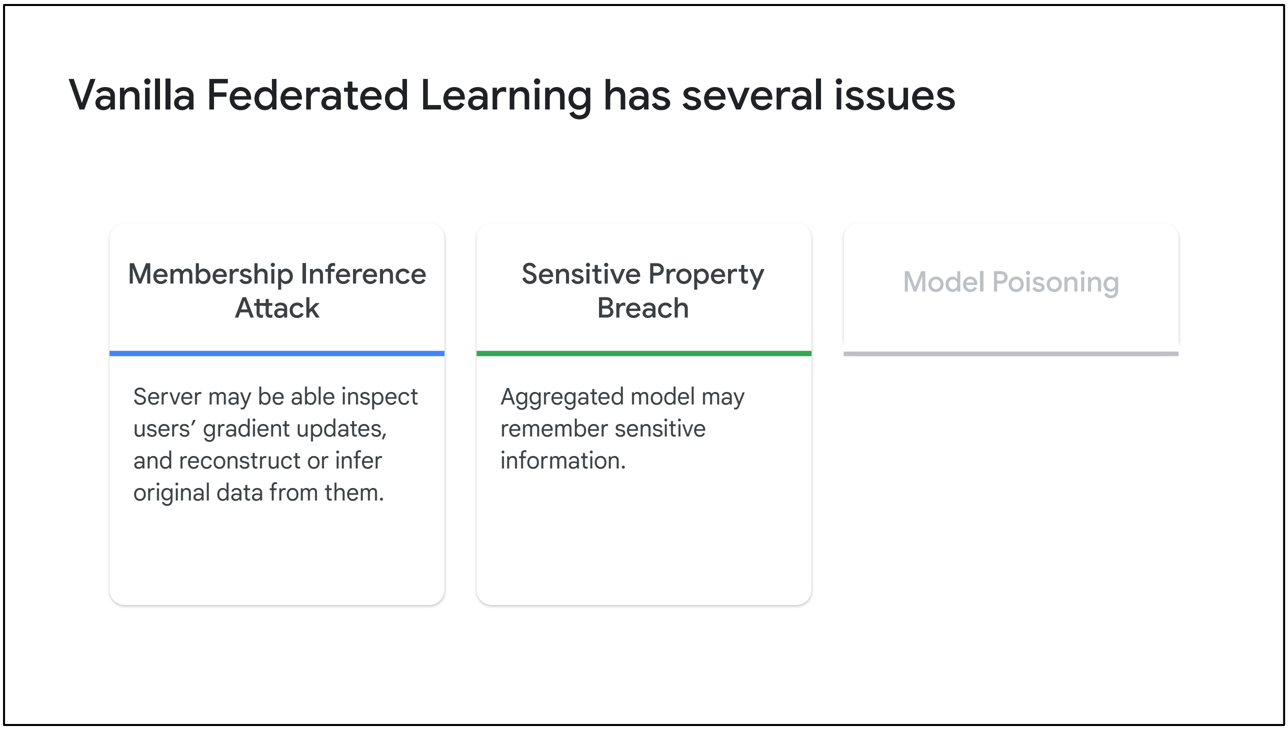
Membership Inference Attack

Server may be able inspect users’ gradient updates, and reconstruct or infer original data from them.

Sensitive Property Breach

Model Poisoning

Membership inference attack: This is a privacy-focused attack against machine learning models. In this attack, an adversary with access to the gradients (or central model provider) tries to determine whether a specific data record (or data point) was used to train the target model based on inferences from gradients.



Vanilla Federated Learning has several issues

Membership Inference Attack

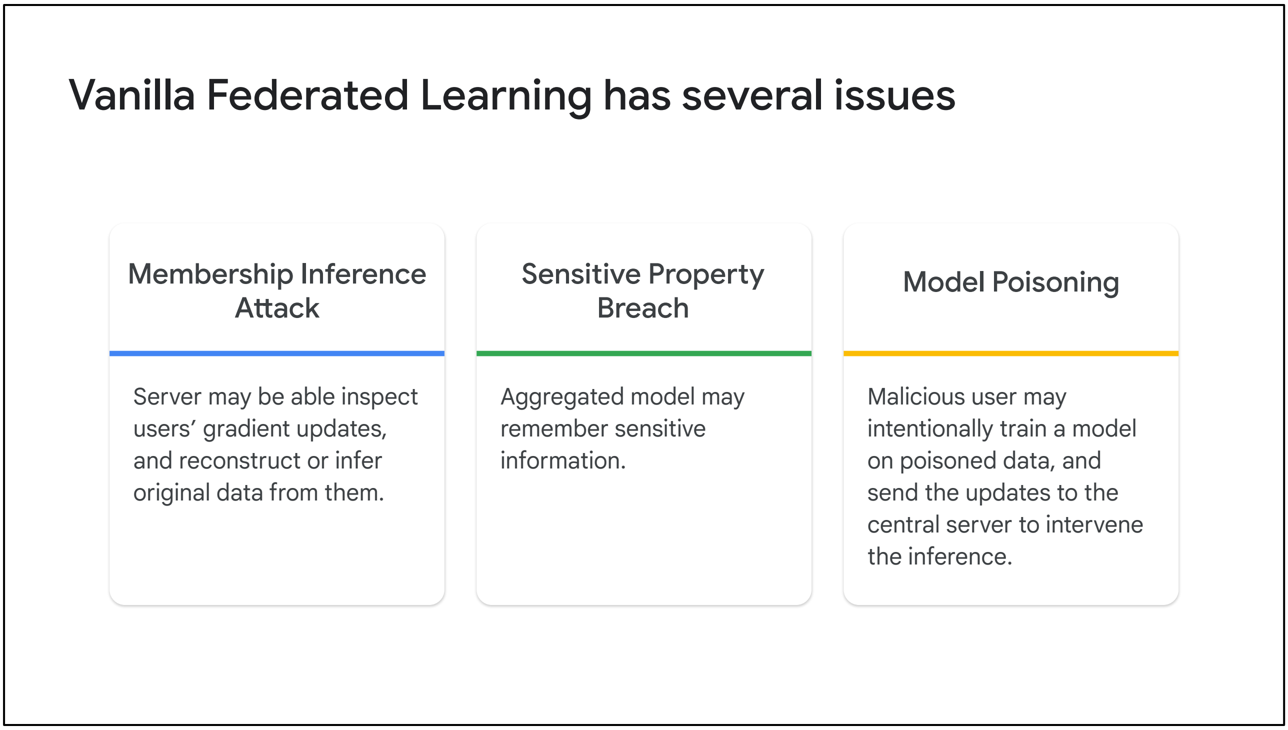
Server may be able inspect users’ gradient updates, and reconstruct or infer original data from them.

Sensitive Property Breach

Aggregated model may remember sensitive information.

Model Poisoning

Sensitive property breach: This is a similar attack to membership inference attack, but involves the exposure of sensitive characteristics or patterns that the model has learned, this exposure comes from reconstructing sensitive features from gradients.



Vanilla Federated Learning has several issues

Membership Inference Attack

Server may be able inspect users’ gradient updates, and reconstruct or infer original data from them.

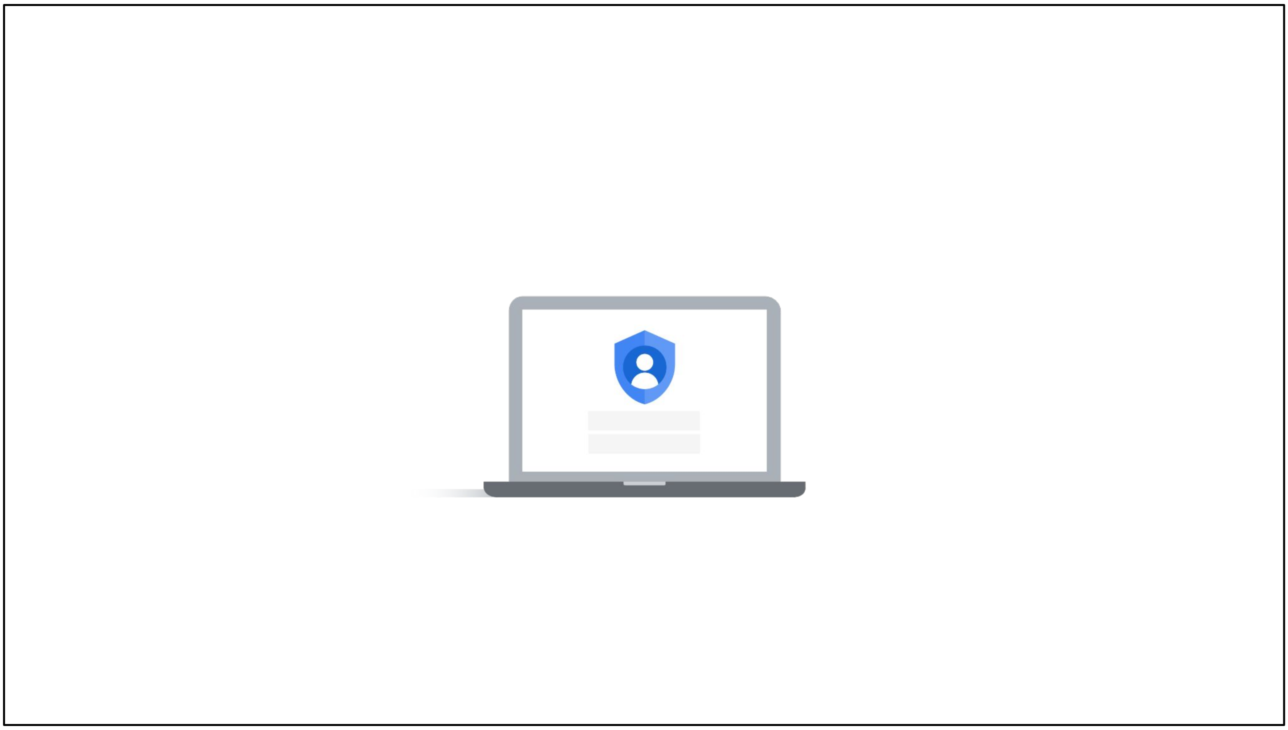
Sensitive Property Breach

Aggregated model may remember sensitive information.

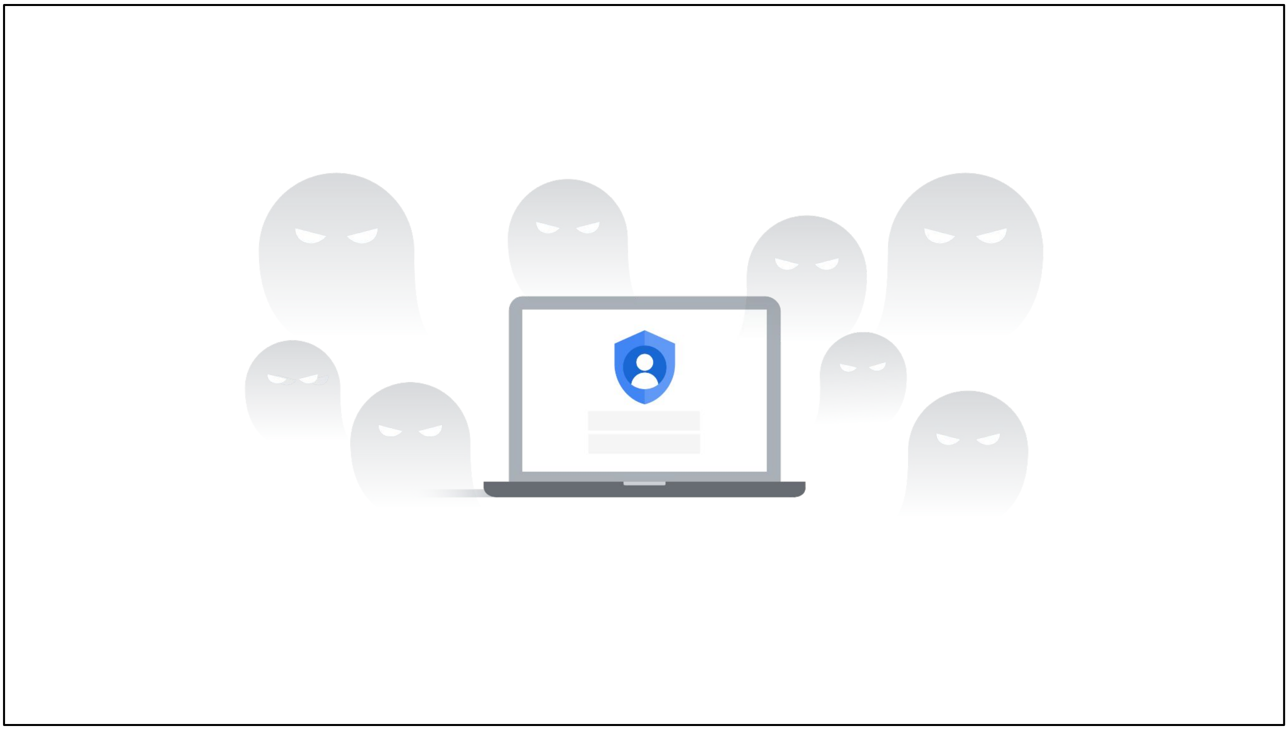
Model Poisoning

Malicious user may intentionally train a model on poisoned data, and send the updates to the central server to intervene the inference.

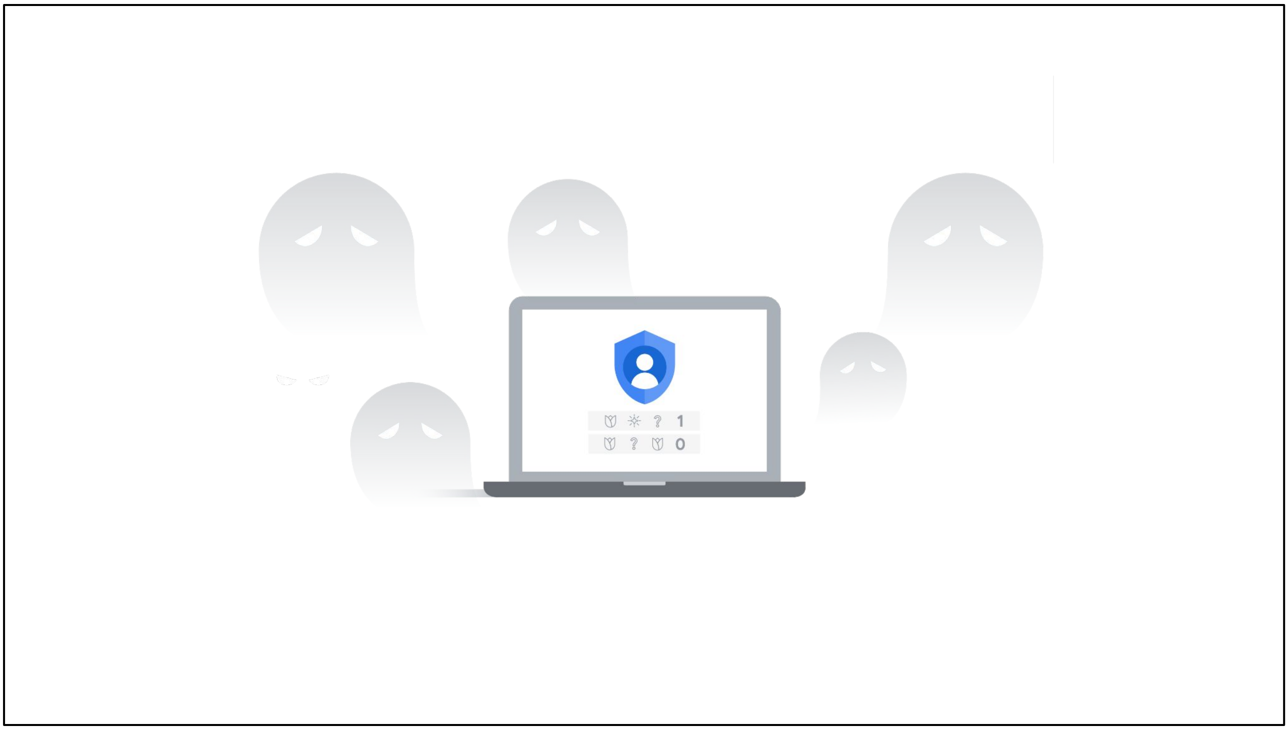
Model poisoning: This occurs when a user who participates in the federated learning process wants to manipulate the model behavior and intentionally creates “poisoned” data points on its device in order to degrade the performance of the model, cause it to make incorrect predictions, or behave in a way that benefits the attacker. Since the provider doesn’t know the data itself, even if they apply anomaly detections on shared gradients, it is difficult to tell if a set of gradients came from a user with malicious intent or from a user with unique behavior.



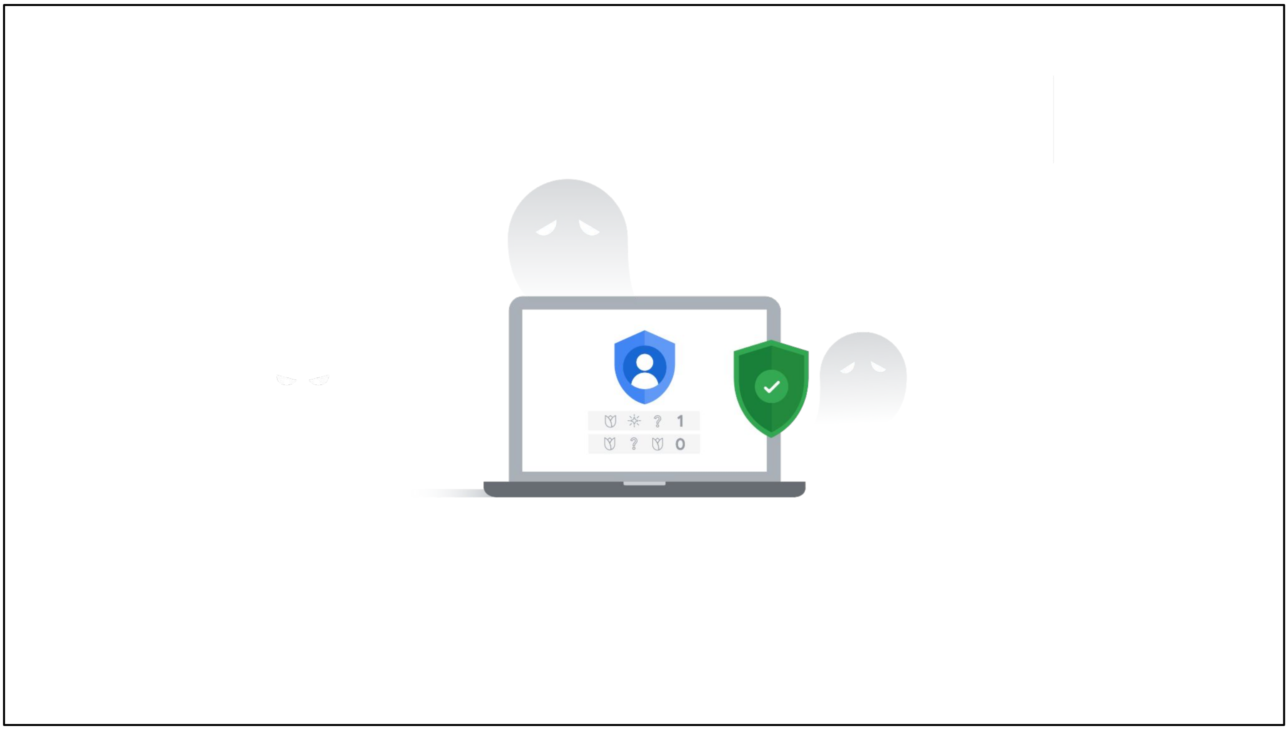
Although there is not a perfect solution that can solve all problems at once,



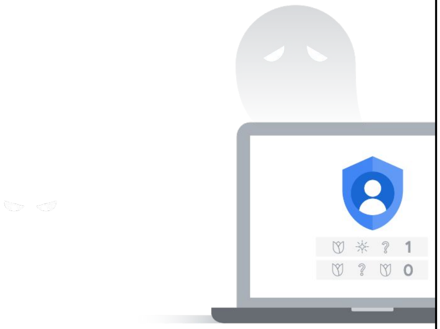
the secure aggregation method can help you avoid membership inference attacks and sensitive property breaches.



Secure aggregation is a cryptographic method used to aggregate multiple data



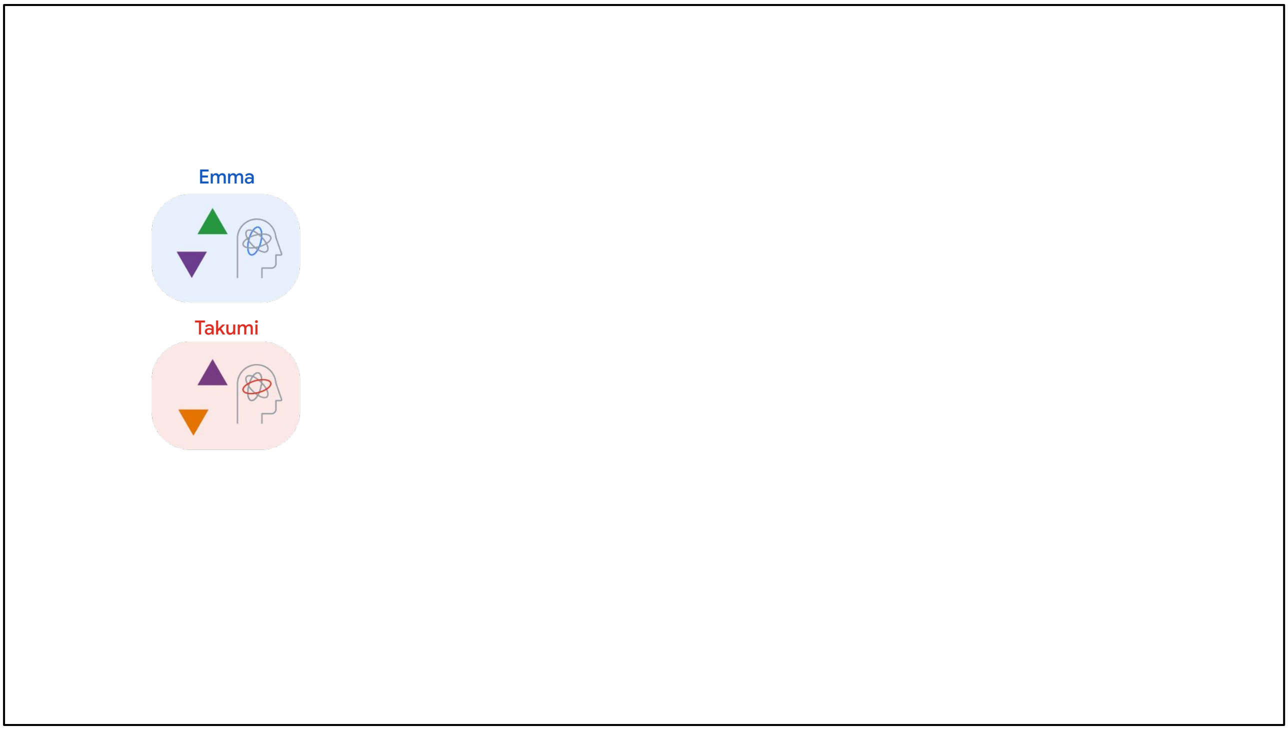
in a secure way by encrypting data before it is sent to the server.


1
0

Emma

Emma

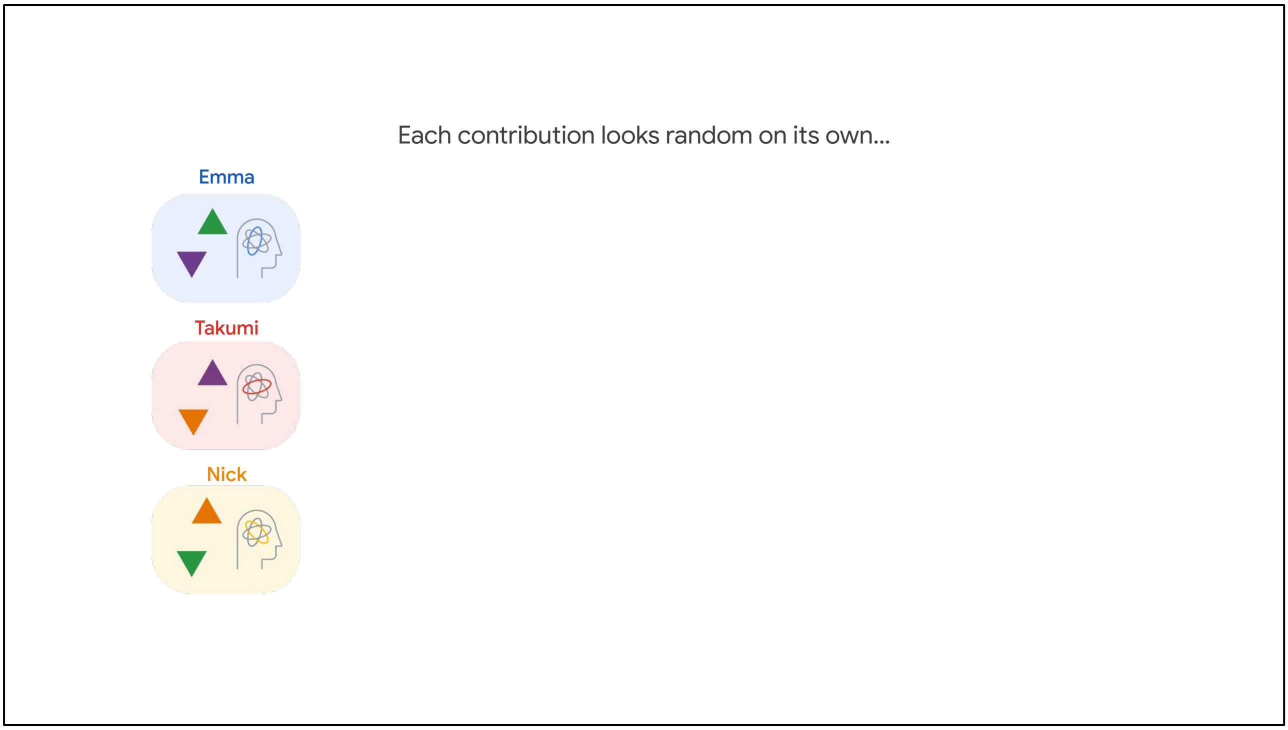
Let’s say we compute a sum of three data points from Emma,



Emma

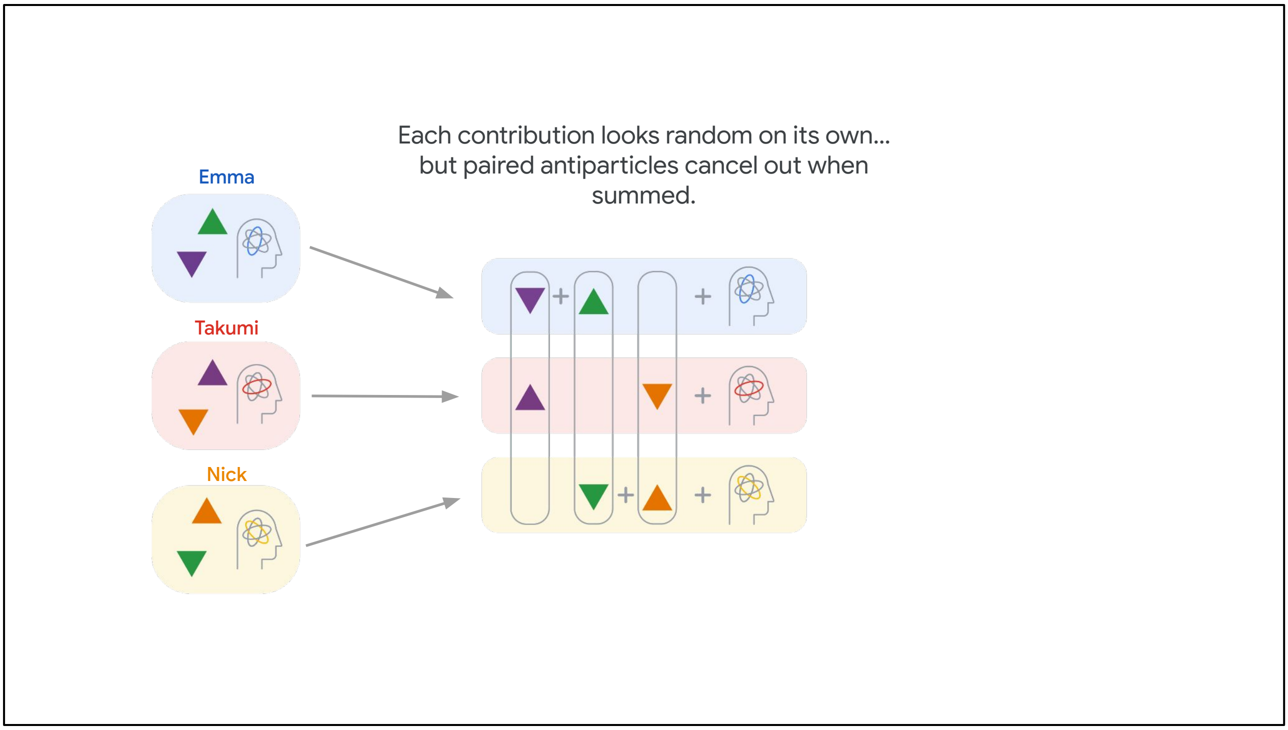
Takumi

Takumi,



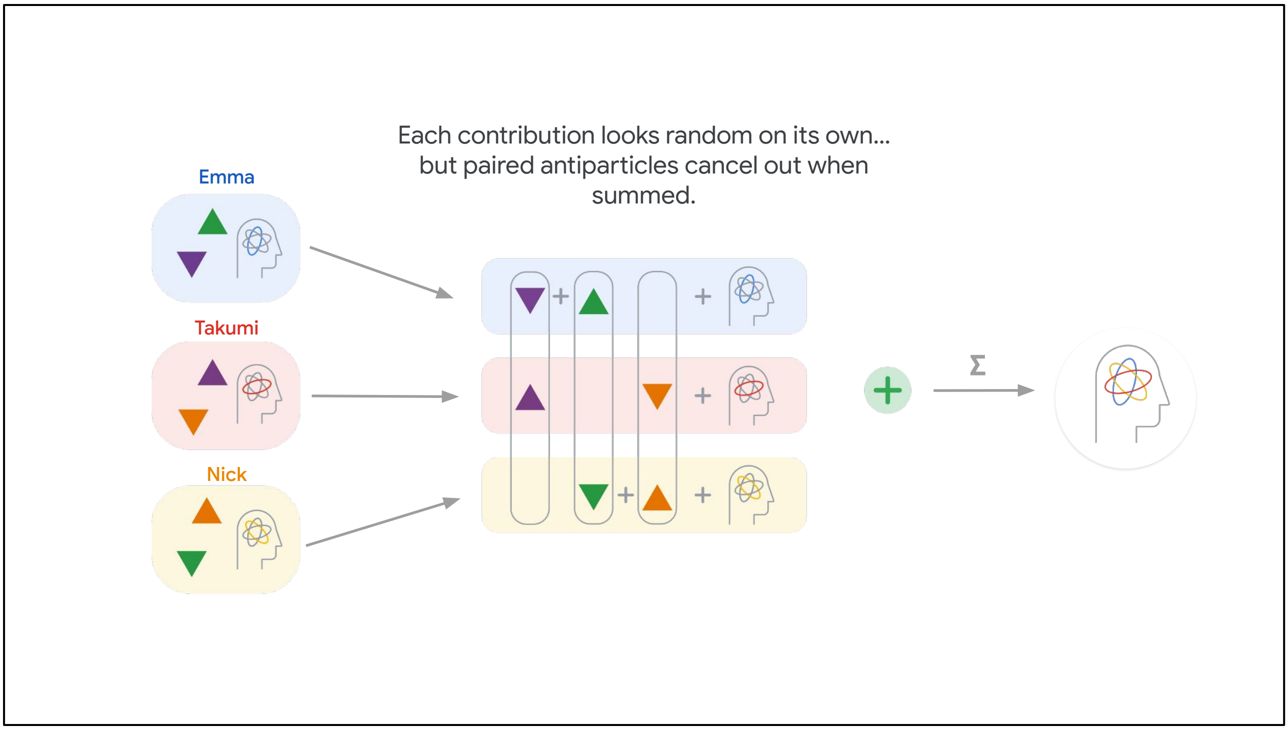
Each contribution looks random on its own...

and Nick. In secure aggregation, instead of simply adding the data up on the central server, the participants modify the values a little by adding a value shared with other participants. The paired two values add up to 0.



Each contribution looks random on its own...
but paired antiparticles cancel out when
summed.

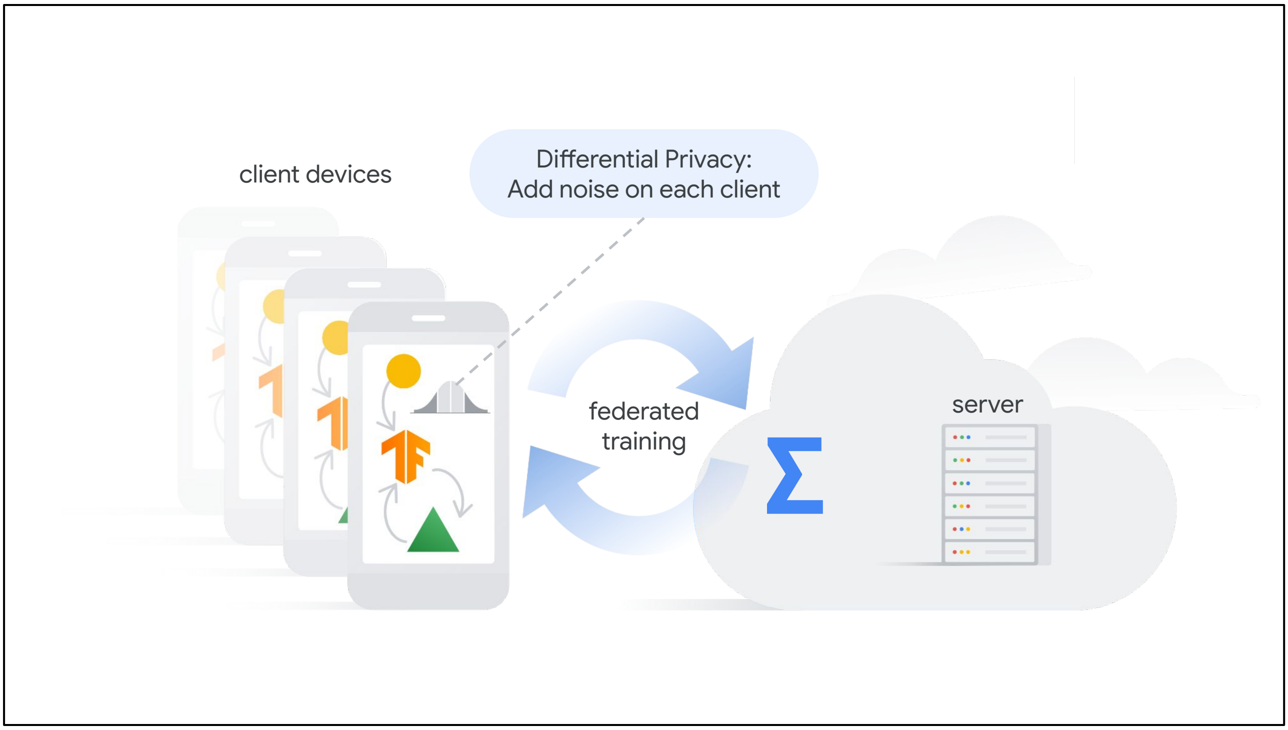
In this case, Emma and Nick share a green value pair, Emma and Takumi share purple, and Takumi and Nick share orange.



Each contribution looks random on its own...
but paired antiparticles cancel out when
summed.

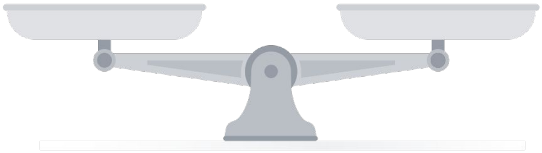
The server receives these perturbed values and aggregates them. The values don't affect the result of aggregation since the pairs add up to 0. At the same time, it is much harder for the server to reconstruct the original data because of the modifications.

By adding secure aggregation, we can apply federated learning in a more secure way.

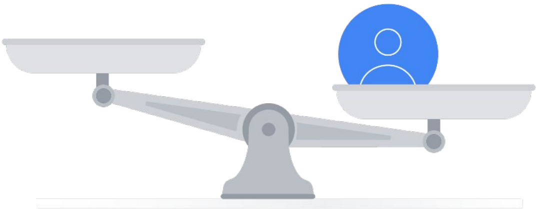


You can also use differential privacy in federated learning as well.
Clients can add noise to their gradients, or the server can take care of it.

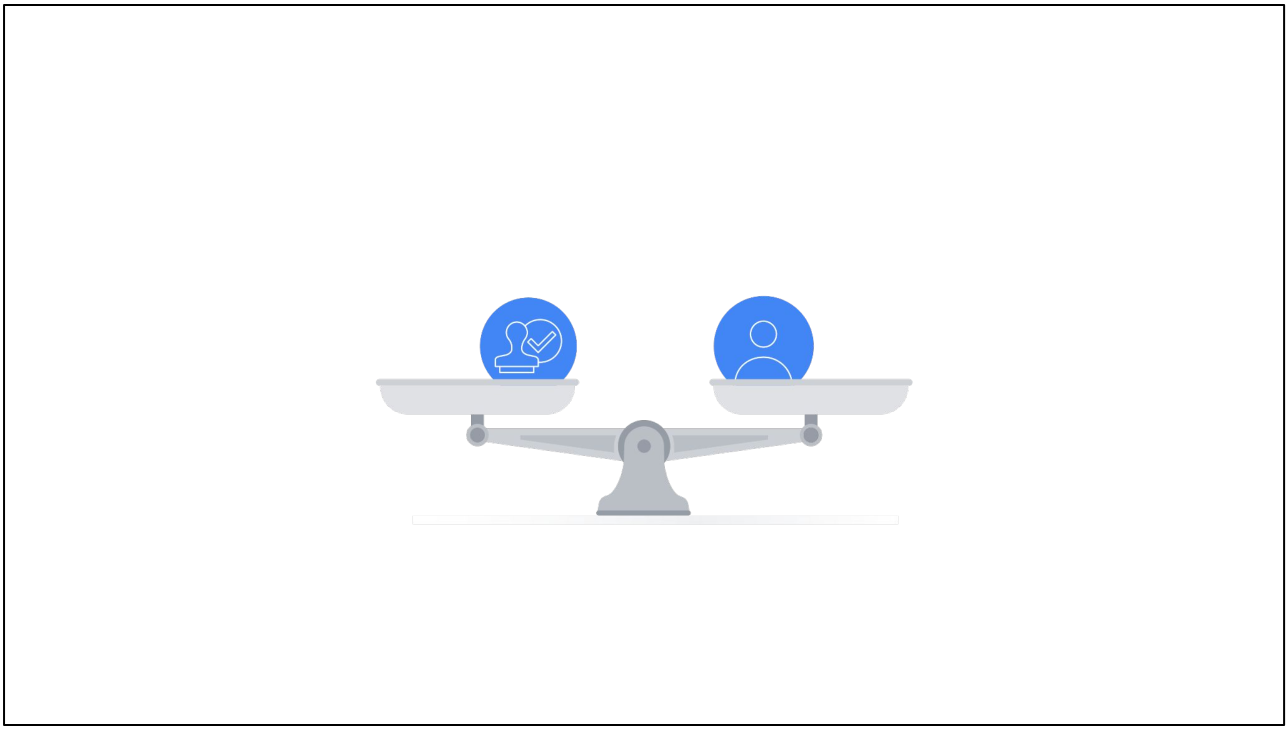
By applying differential privacy in federated learning, you mitigate the risk of gradients revealing sensitive information of a specific user. In addition, you can make a more robust update strategy to avoid data poisoning.



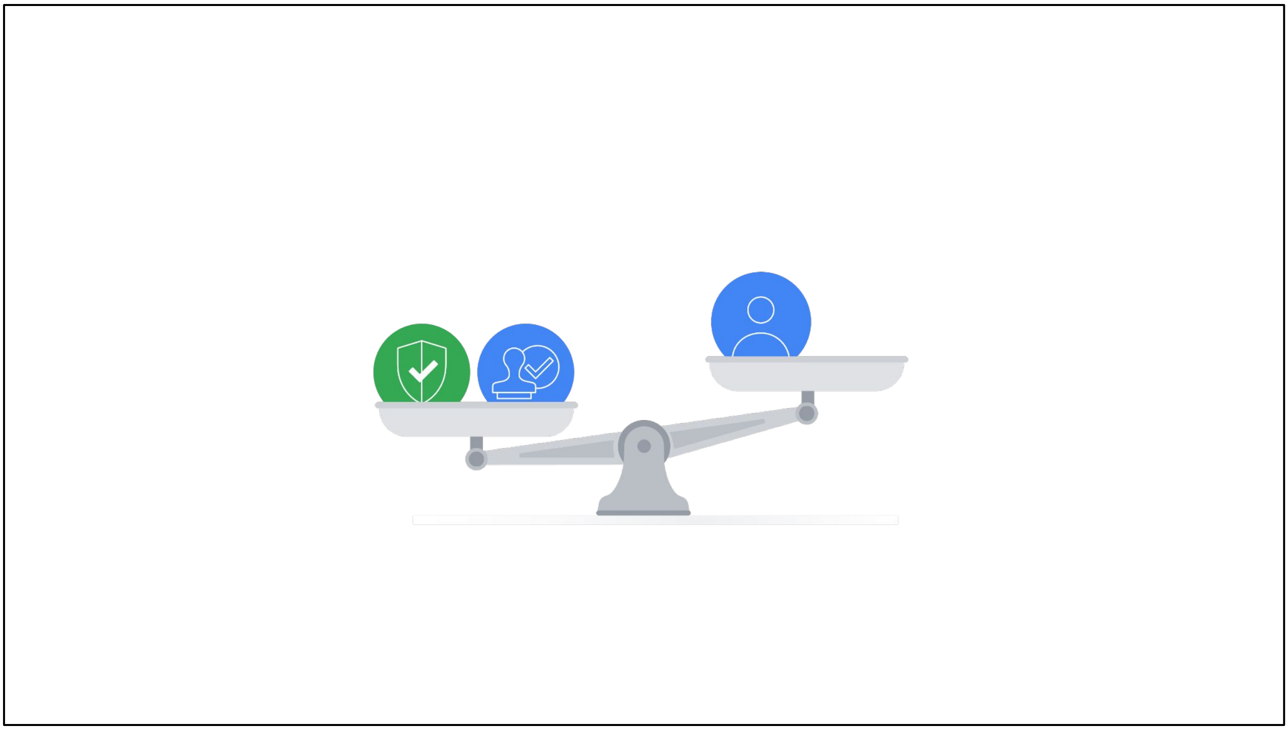
However, you should note that there is no perfect solution and each technique has tradeoffs between multiple factors



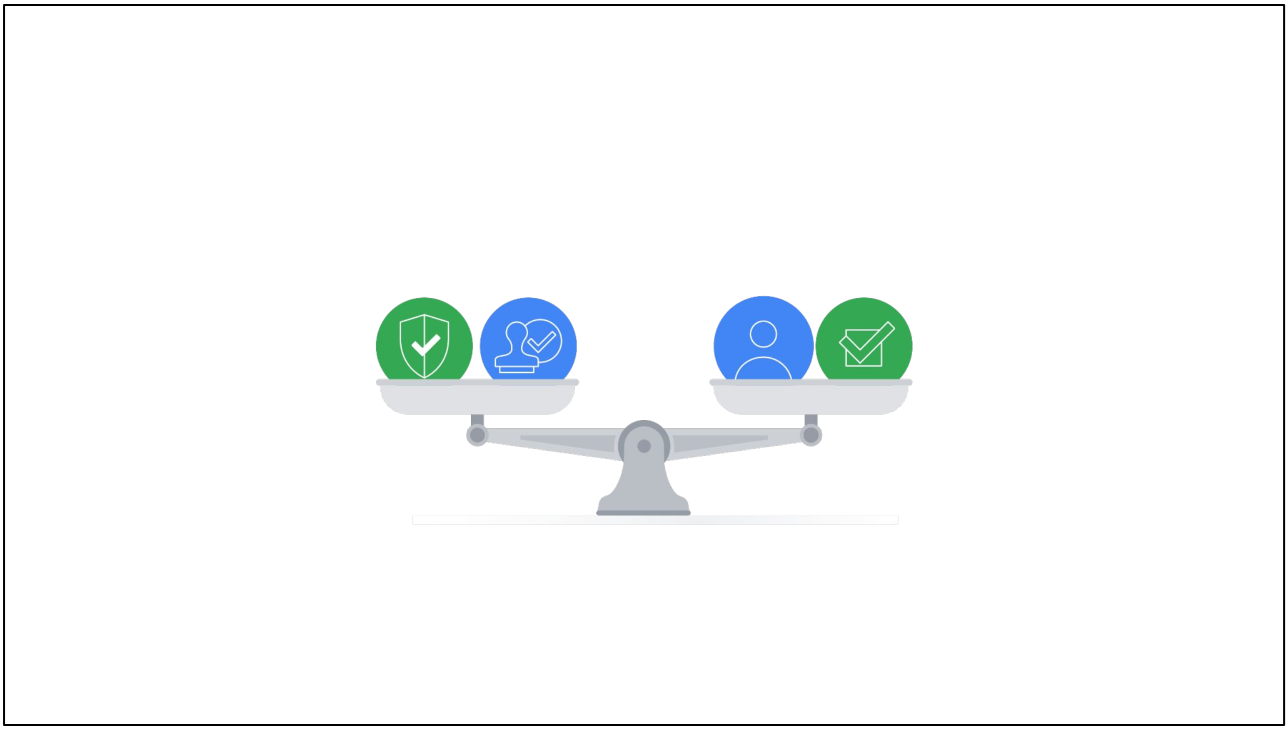
such as privacy,



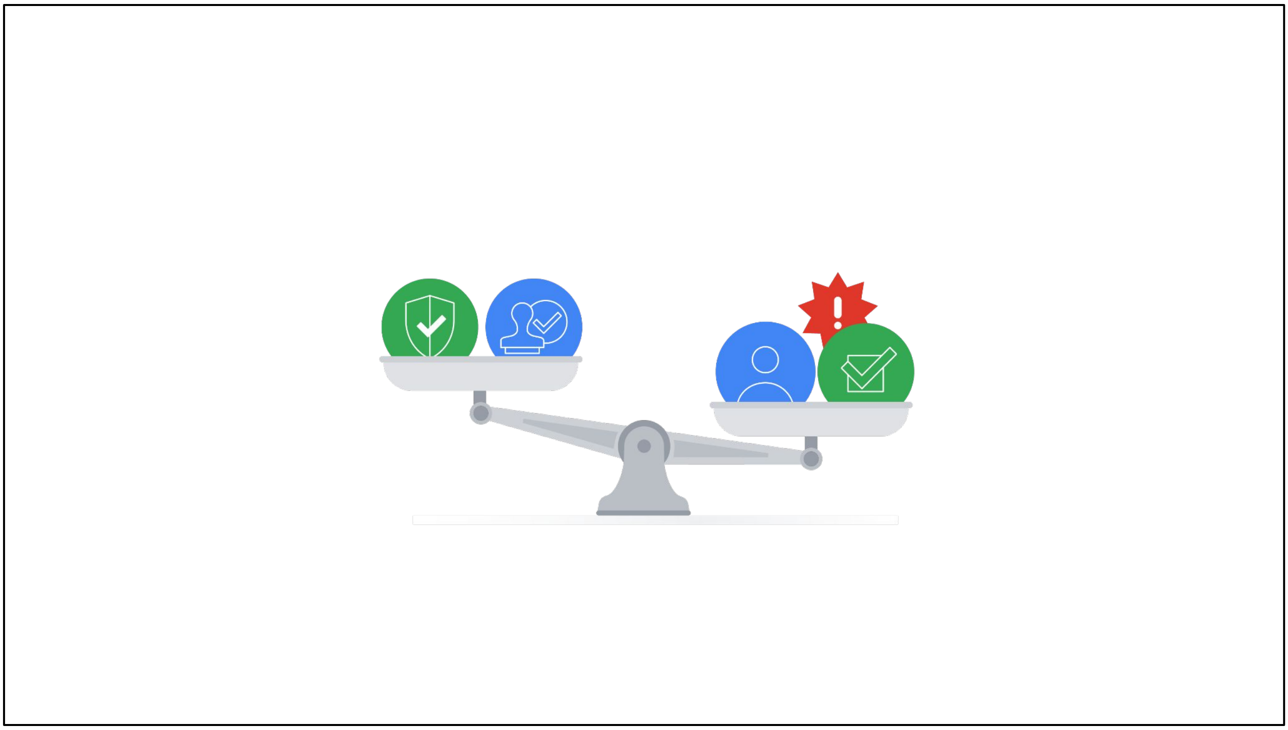
performance,



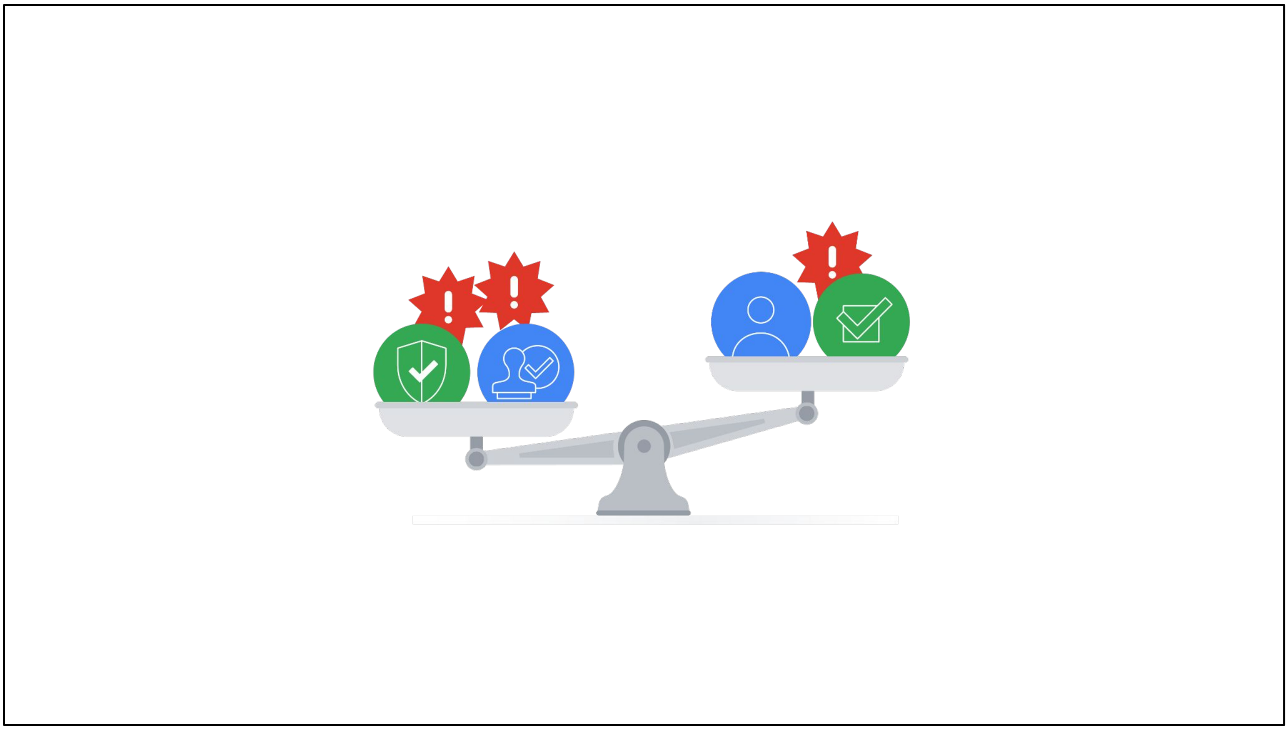
robustness,



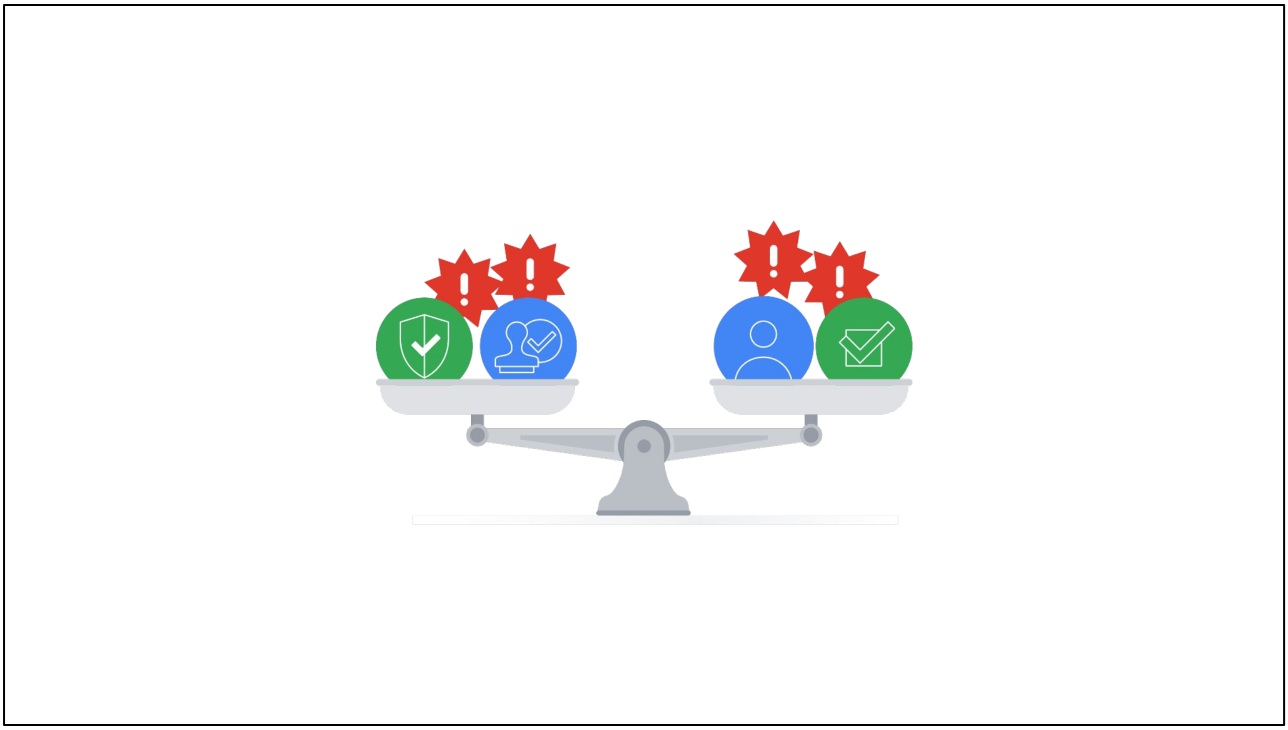
and fairness.



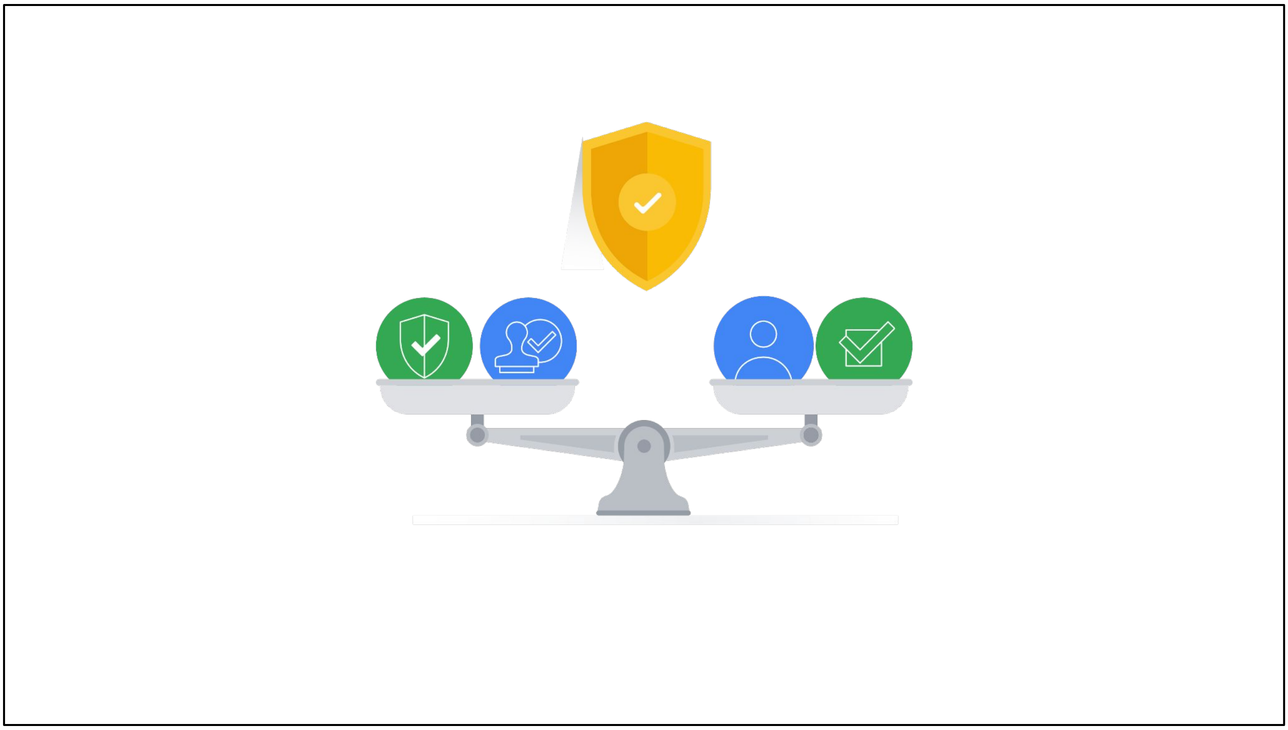
If you apply large noise using differential privacy in order to provide stronger privacy, it may affect performance.



And, although differential privacy can make the process robust, it could mean the system deprioritized data from users that were unique or in the minority compared to the rest of the data. This could potentially lead to a bias issue.



In summary, to ensure stronger privacy in machine learning models, it is important to understand the requirements of a system,



then evaluate the system using multiple proper metrics and criteria.

01 Overview of AI Privacy

02 Privacy in Training Data: De-identification techniques

03 Privacy in Training Data: Randomization techniques

04 Privacy in Machine Learning Training: DP-SGD

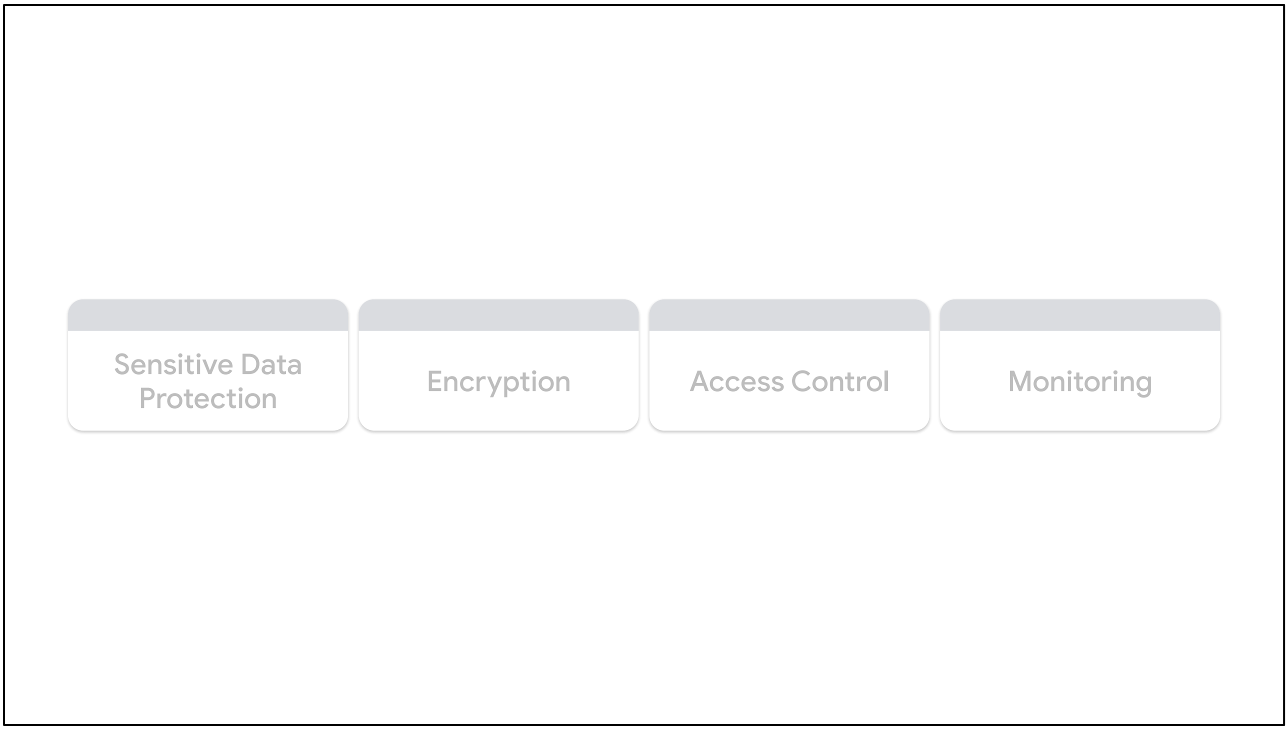
05 Privacy in Machine Learning Training: Federated Learning

06 System Security on Google Cloud

07 System Security and Generative AI

08 Lab: Differential Privacy in Machine Learning with TensorFlow Privacy

So far, we have looked at privacy in data and model training. Let’s take some time now to look at system security on Google Cloud.



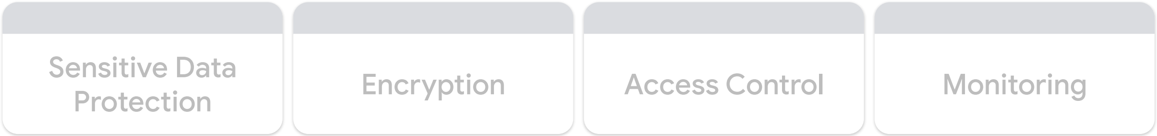
Sensitive Data Protection

Encryption

Access Control

Monitoring

In addition to thinking about data and model training privacy considerations, it is important to follow the standard system security best practices, including access control and monitoring.

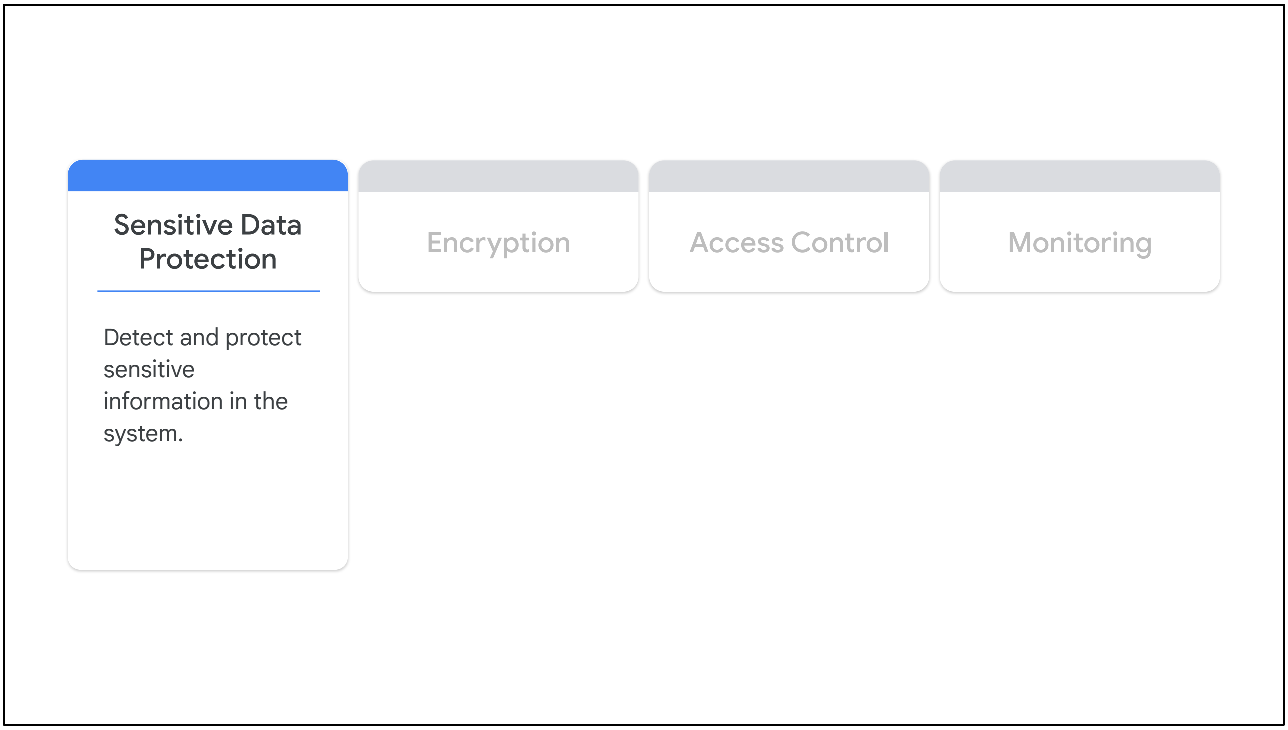


Sensitive Data Protection

Encryption

Access Control

Monitoring



Sensitive Data Protection

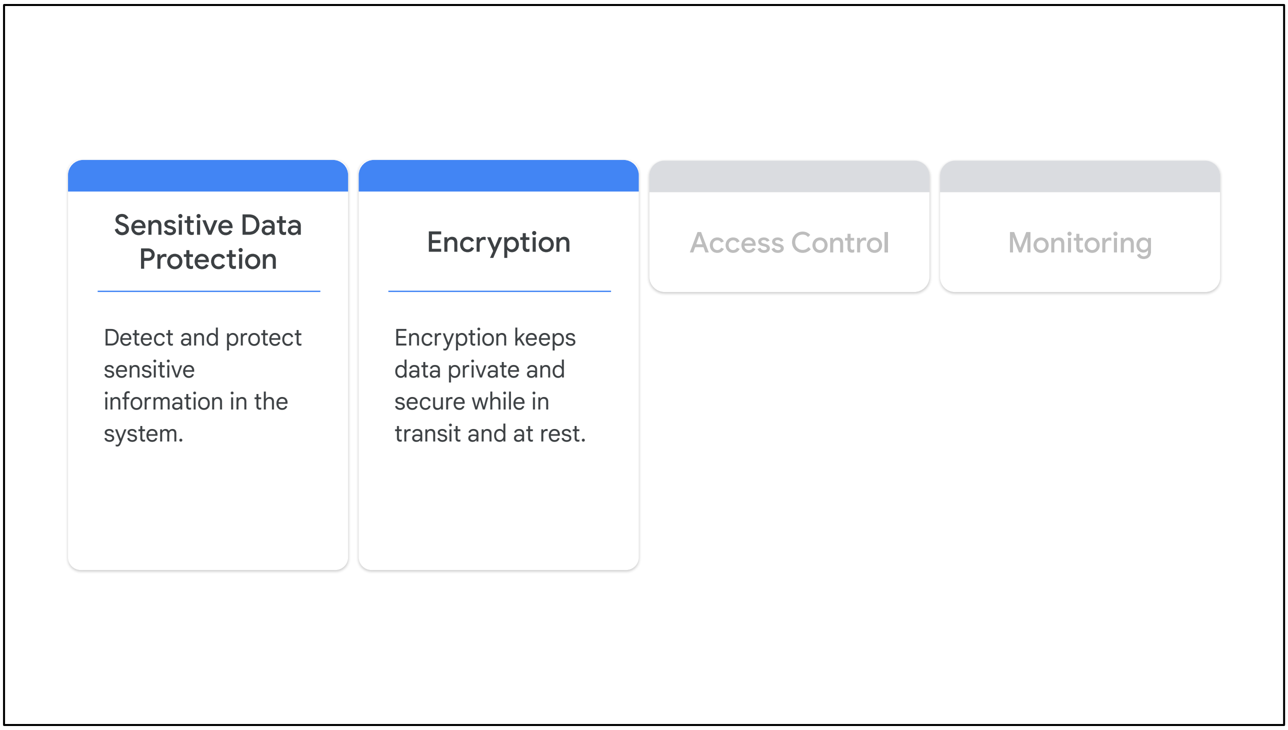
Detect and protect sensitive information in the system.

Encryption

Access Control

Monitoring

- Sensitive data protection: Incorporate a system that automatically detects and prevents online threats so that you can be confident that private information is safe.



Sensitive Data Protection

Detect and protect sensitive information in the system.

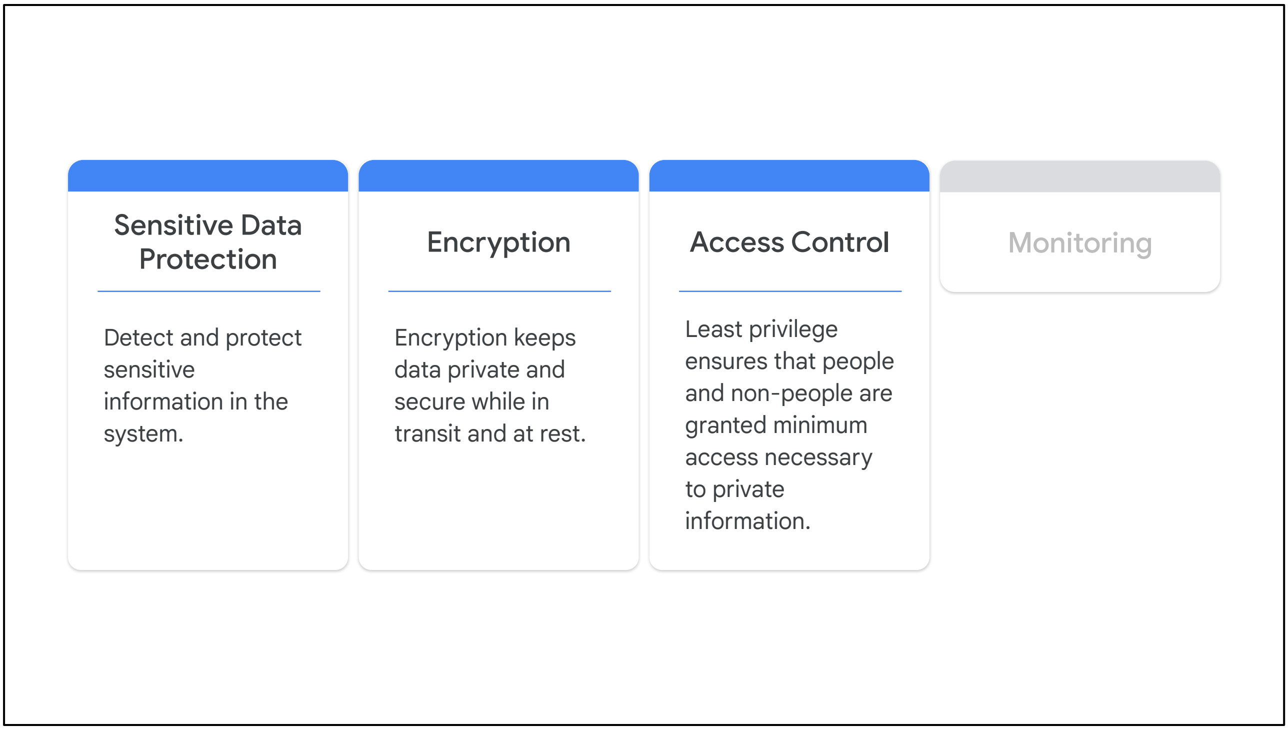
Encryption

Encryption keeps data private and secure while in transit and at rest.

Access Control

Monitoring

- Encryption: Keep data private and secure while in transit and at rest.



Sensitive Data Protection

Detect and protect sensitive information in the system.

Encryption

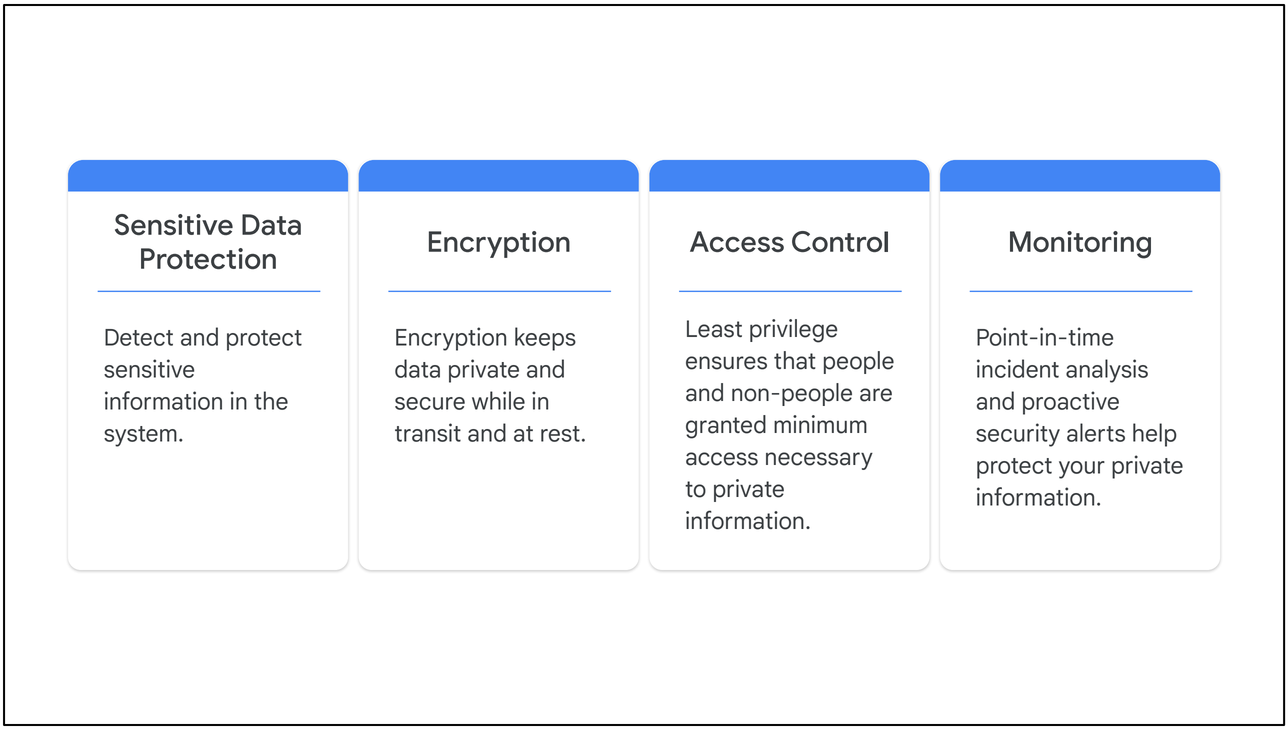
Encryption keeps data private and secure while in transit and at rest.

Access Control

Least privilege ensures that people and non-people are granted minimum access necessary to private information.

Monitoring

- Access control: Follow least-privilege access configurations. You can ensure that people and non-people are granted minimum access necessary to private information.



Sensitive Data Protection

Detect and protect sensitive information in the system.

Encryption

Encryption keeps data private and secure while in transit and at rest.

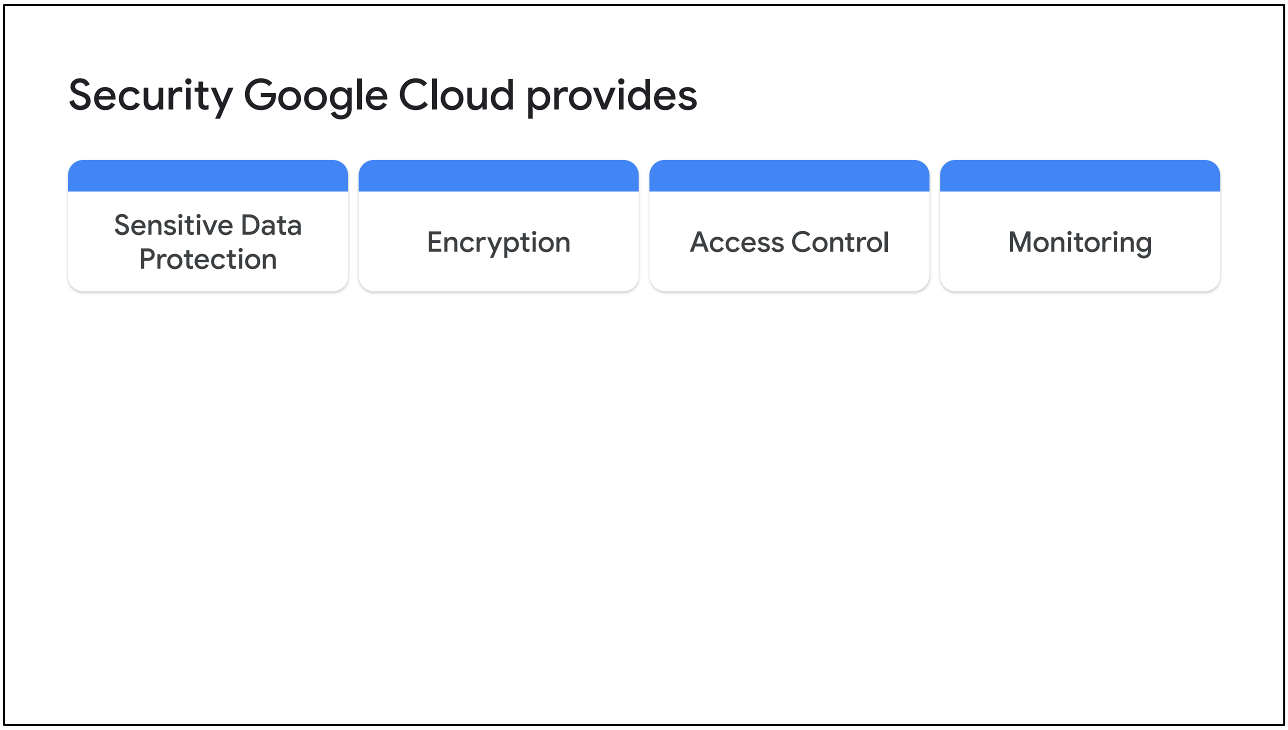
Access Control

Least privilege ensures that people and non-people are granted minimum access necessary to private information.

Monitoring

Point-in-time incident analysis and proactive security alerts help protect your private information.

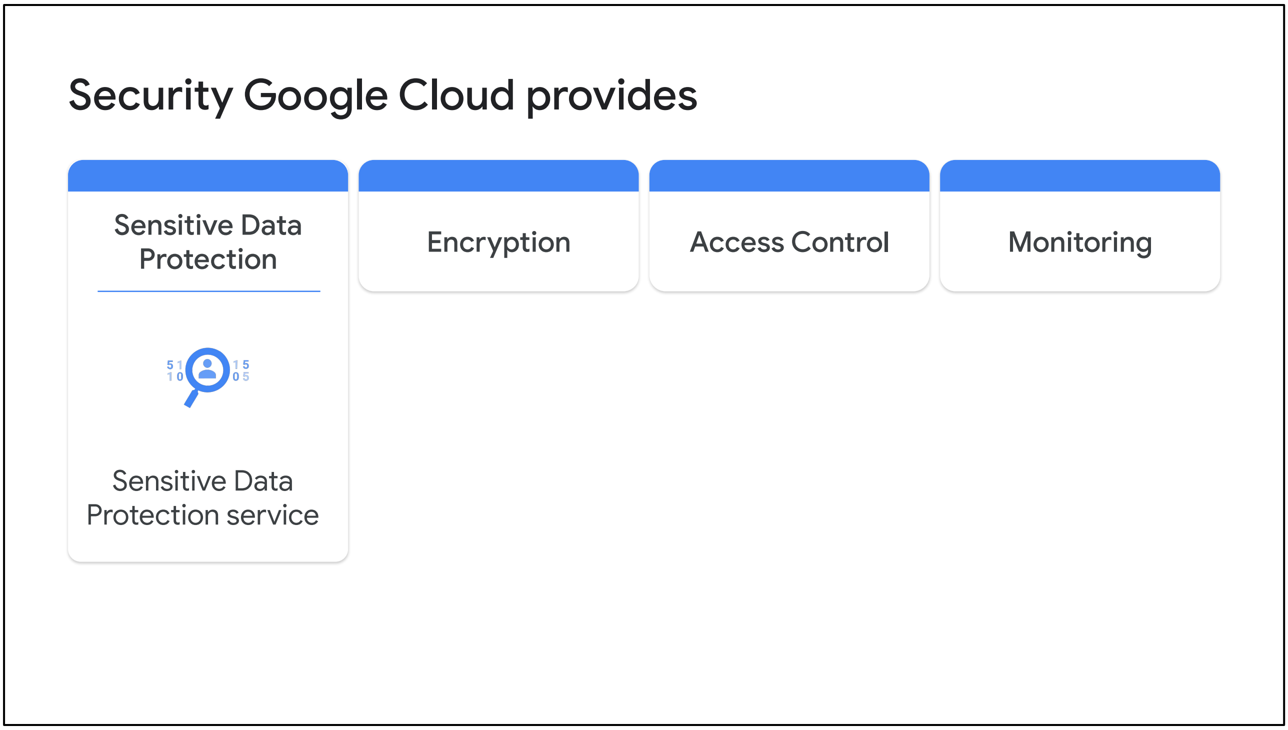
- Monitoring: Implement point-in-time incident analysis and proactive security alerts to help protect private information by reacting promptly to unexpected behavior.



Security Google Cloud provides

- Sensitive Data Protection
- Encryption
- Access Control
- Monitoring

Google Cloud provides a very comprehensive security offering.



Security Google Cloud provides

Sensitive Data Protection

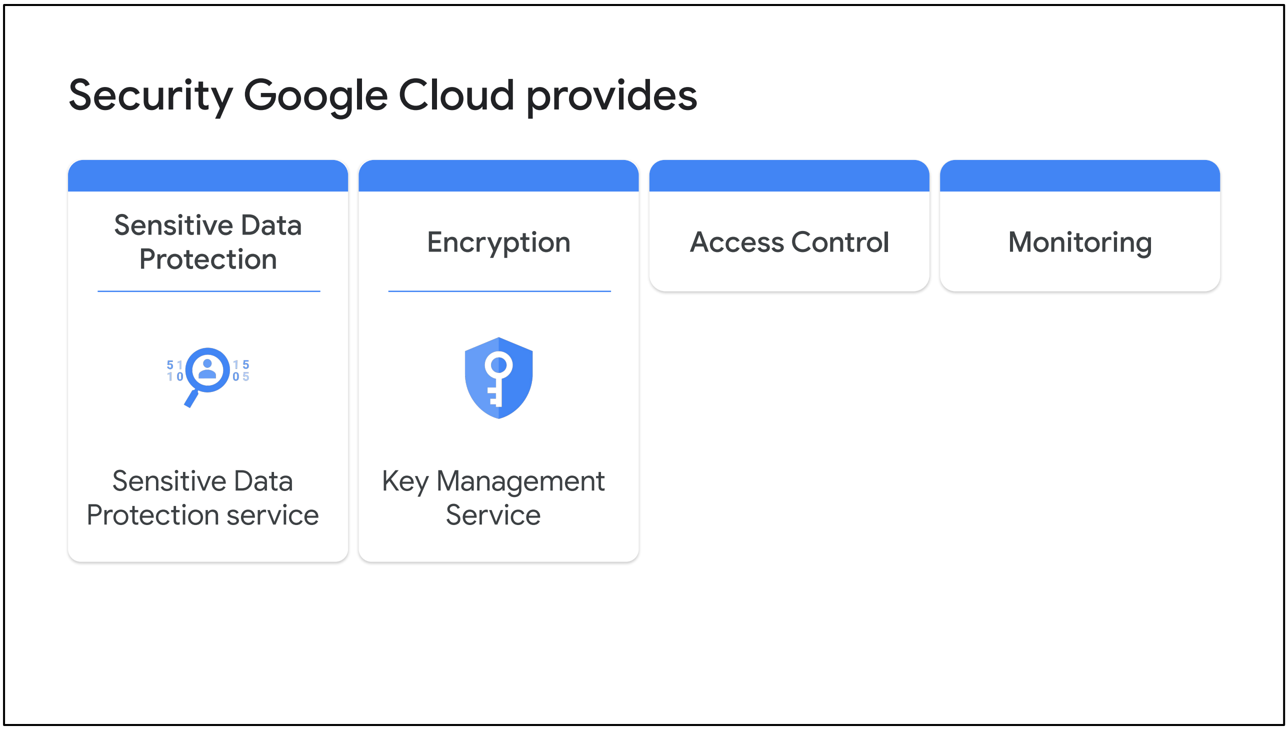
Encryption

Access Control

Monitoring

Sensitive Data Protection service

Sensitive data protection is offered through the Cloud Data Loss Prevention (DLP) API,



Security Google Cloud provides

Sensitive Data Protection

Sensitive Data Protection service

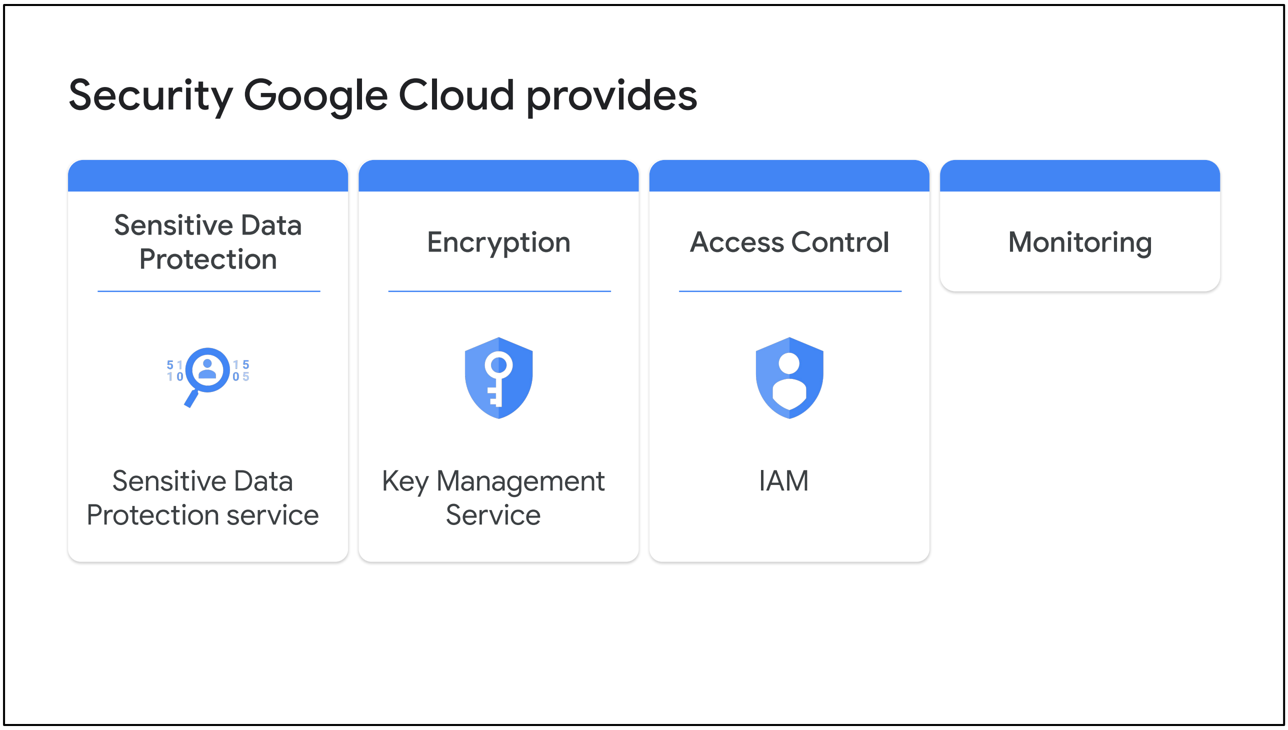
Encryption

Key Management Service

Access Control

Monitoring

encryption through Cloud Key Management Service (or KMS),



Security Google Cloud provides

Sensitive Data Protection

Sensitive Data Protection service

Encryption

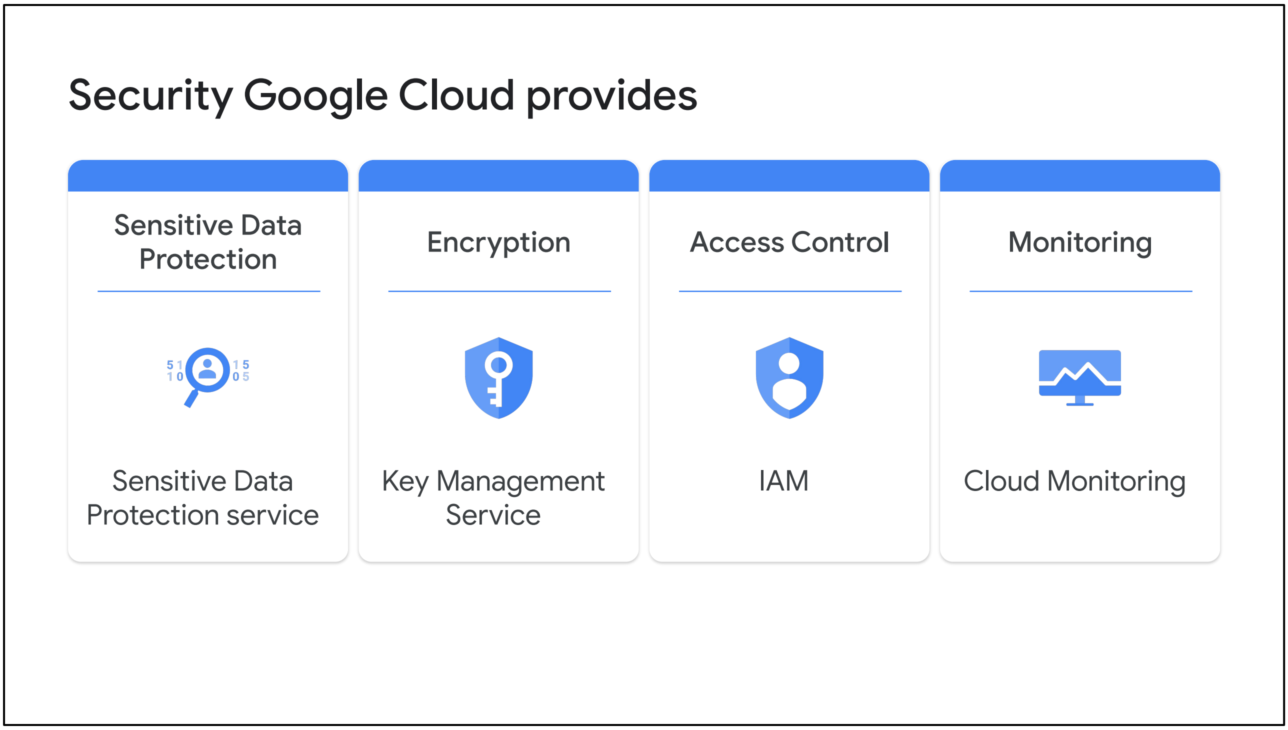
Key Management Service

Access Control

IAM

Monitoring

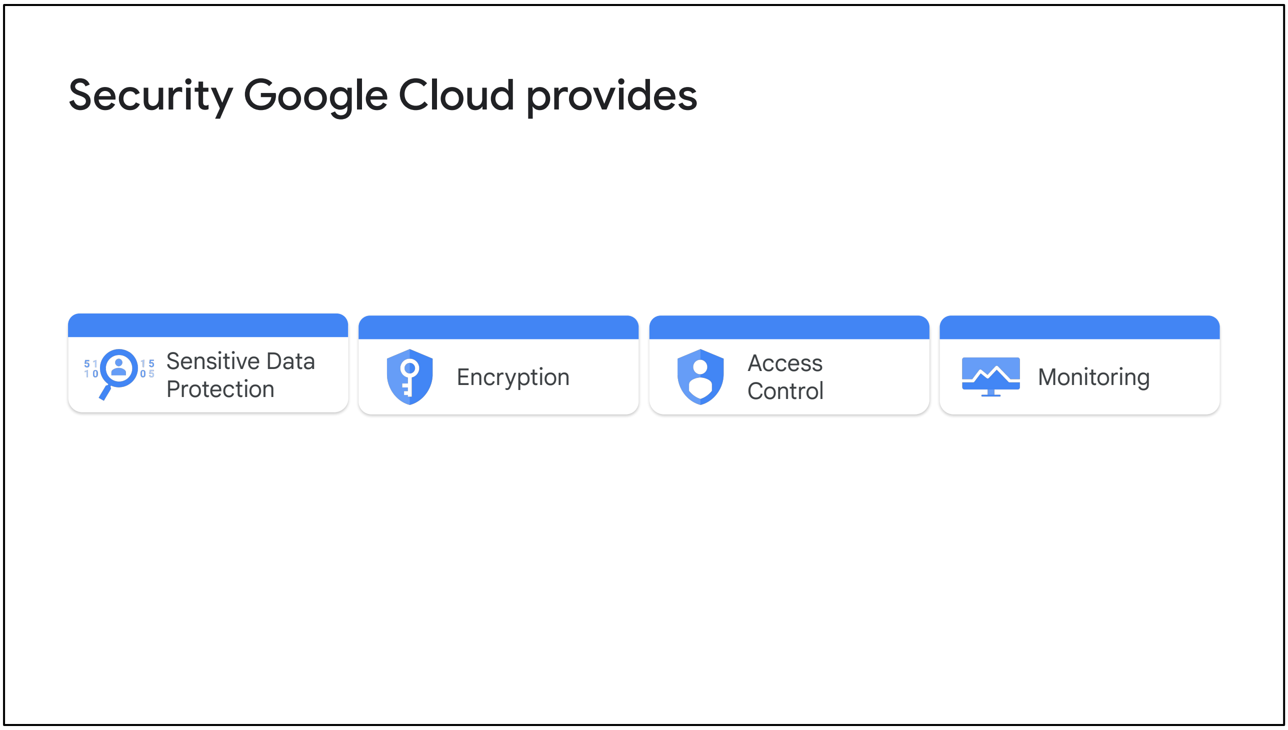
access control through Identity & Access Management (or IAM),



Security Google Cloud provides

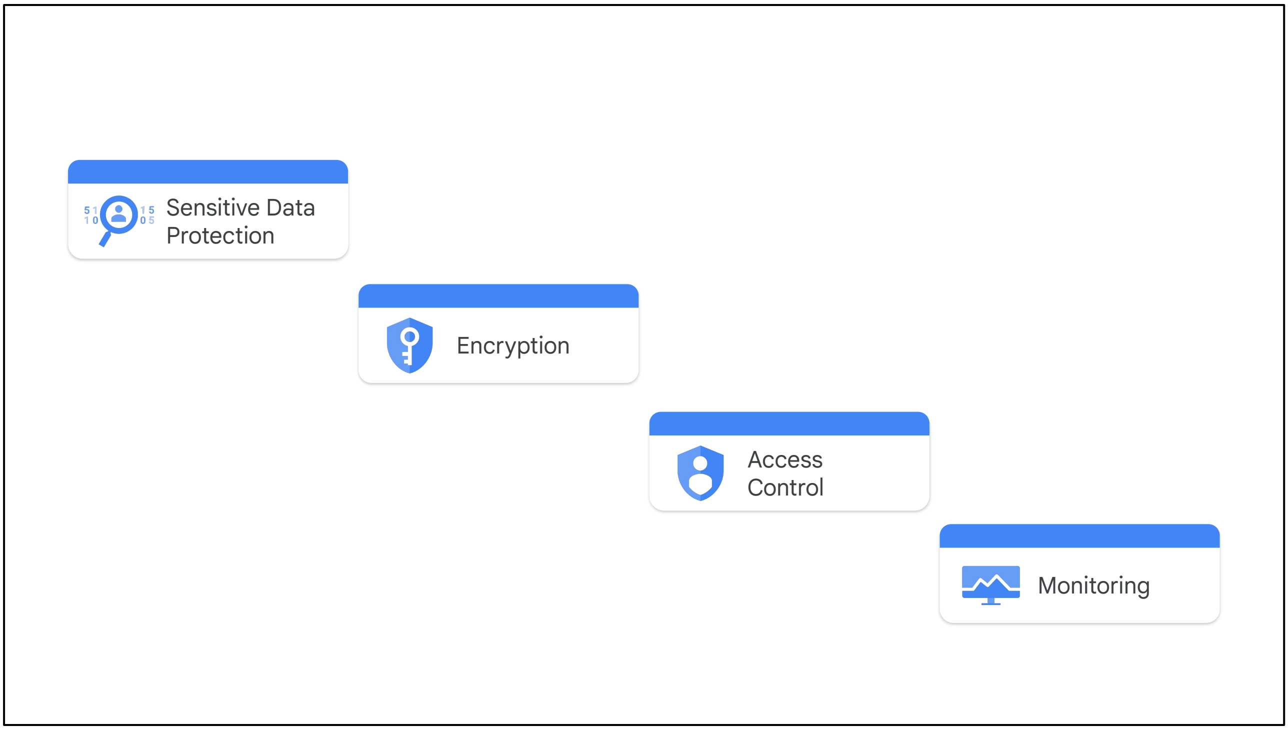
Sensitive Data Protection	Encryption	Access Control	Monitoring
Sensitive Data Protection service	Key Management Service	IAM	Cloud Monitoring

and monitoring via Cloud Monitoring.

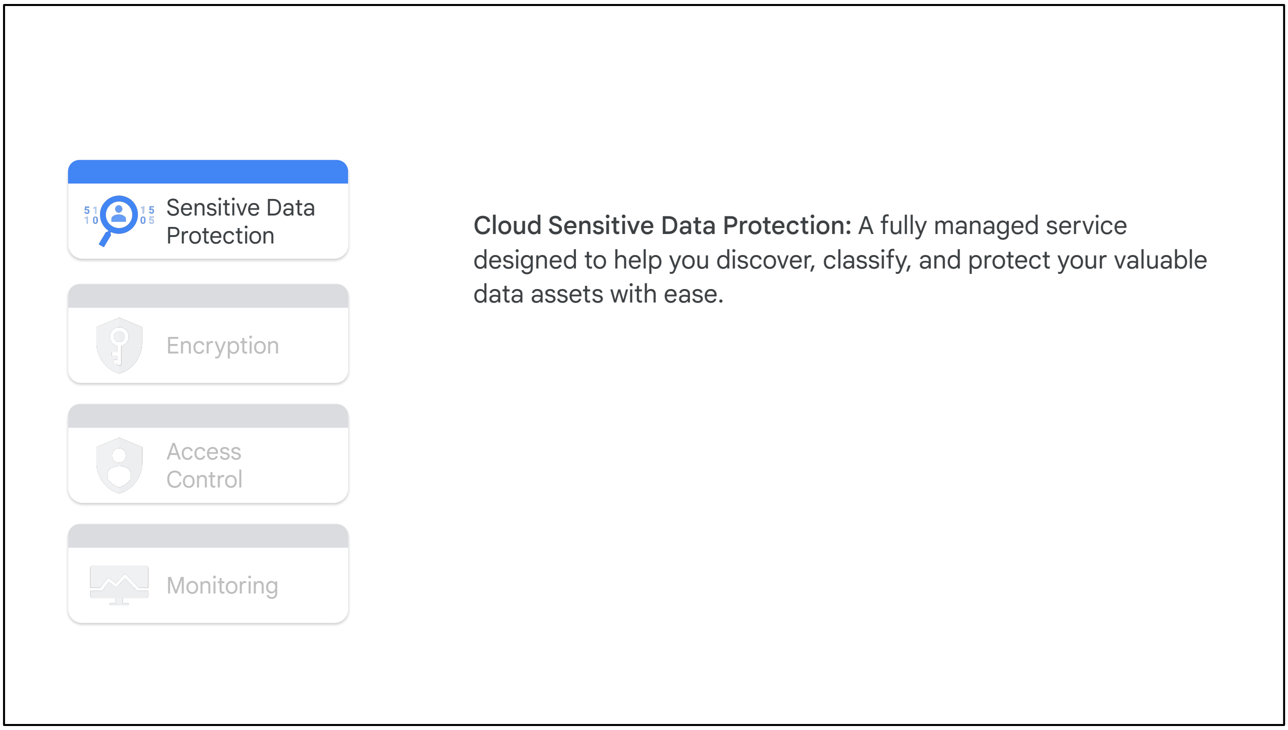


Security Google Cloud provides

- Sensitive Data Protection
- Encryption
- Access Control
- Monitoring


Sensitive Data Protection
Encryption
Access Control
Monitoring

- Sensitive Data Protection
- Encryption
- Access Control
- Monitoring



Sensitive Data Protection

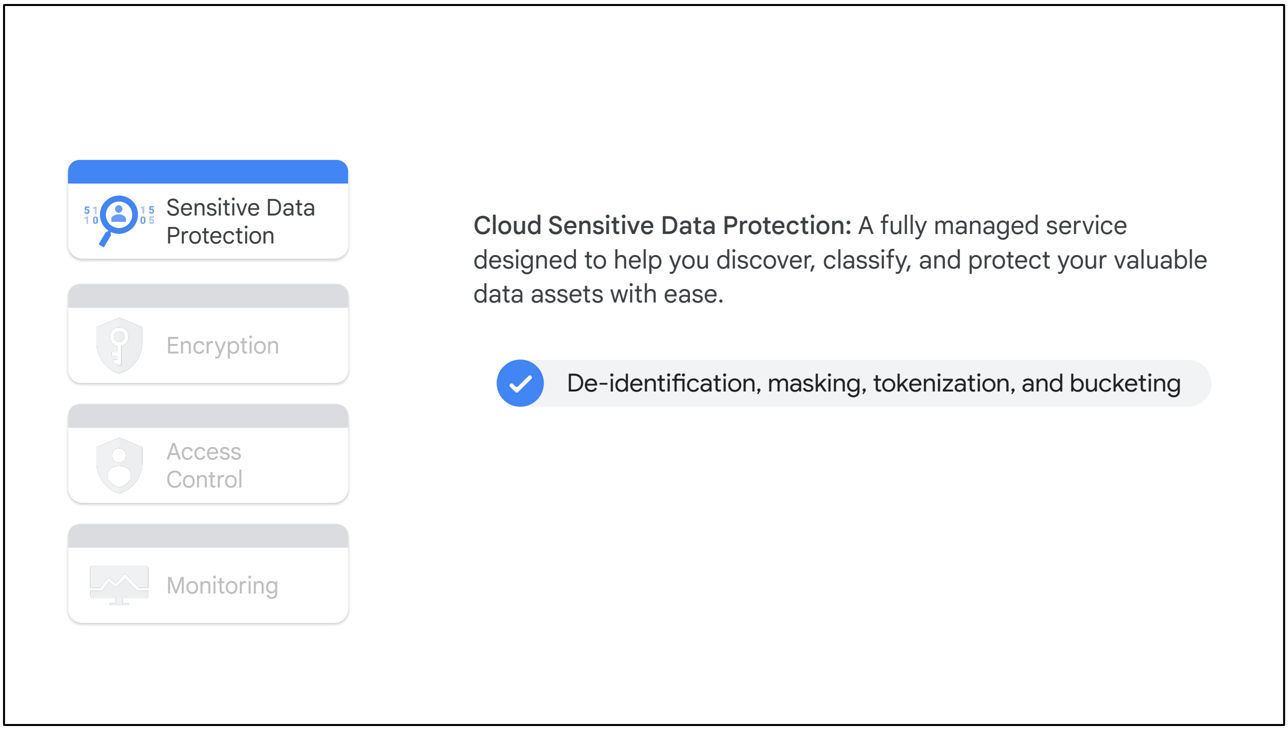
Encryption

Access Control

Monitoring

Cloud Sensitive Data Protection: A fully managed service designed to help you discover, classify, and protect your valuable data assets with ease.

Sensitive data protection through the Cloud Data Loss Prevention service helps you discover, govern, protect, and report on sensitive data across your ecosystem.
The service provides:



Sensitive Data Protection

Encryption

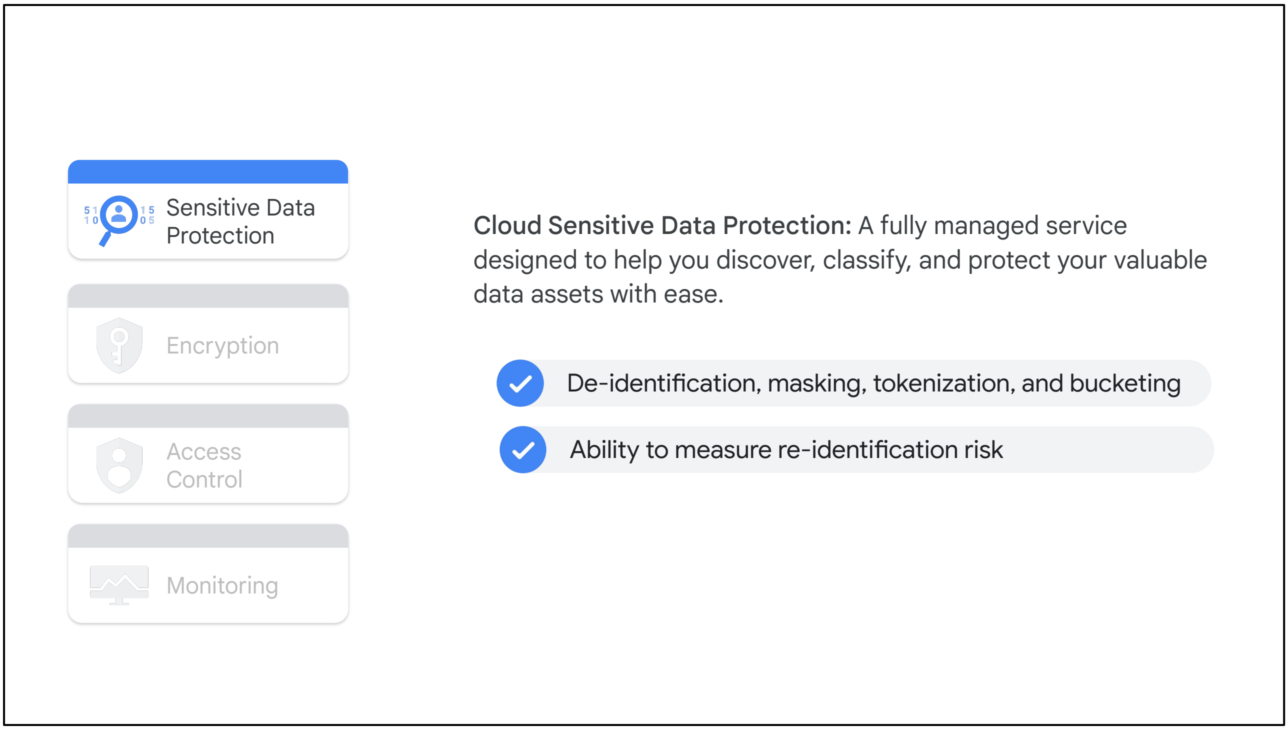
Access Control

Monitoring

Cloud Sensitive Data Protection: A fully managed service designed to help you discover, classify, and protect your valuable data assets with ease.

✓ De-identification, masking, tokenization, and bucketing

- De-identification techniques such as masking and tokenization, with support for structured and unstructured data.



Sensitive Data Protection

Encryption

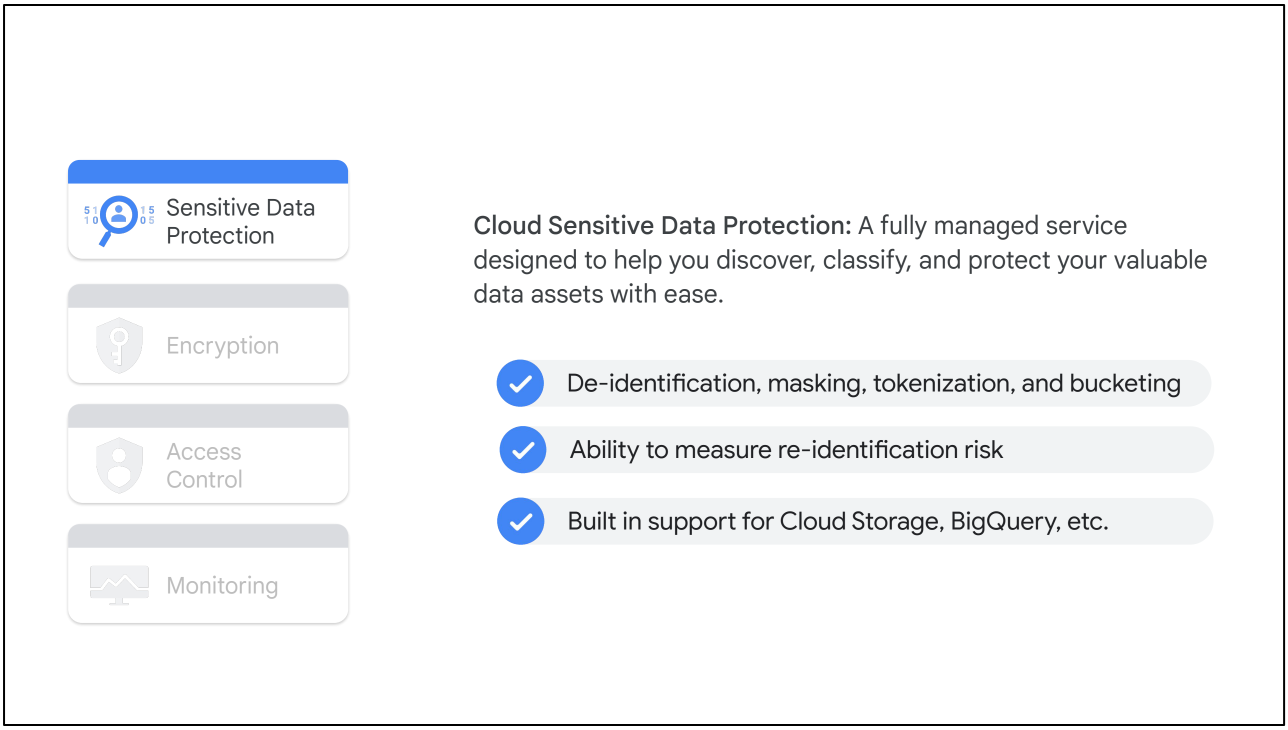
Access Control

Monitoring

Cloud Sensitive Data Protection: A fully managed service designed to help you discover, classify, and protect your valuable data assets with ease.

- ✓ De-identification, masking, tokenization, and bucketing
- ✓ Ability to measure re-identification risk

- Ability to measure re-identification risk in structured data.



Sensitive Data Protection

Encryption

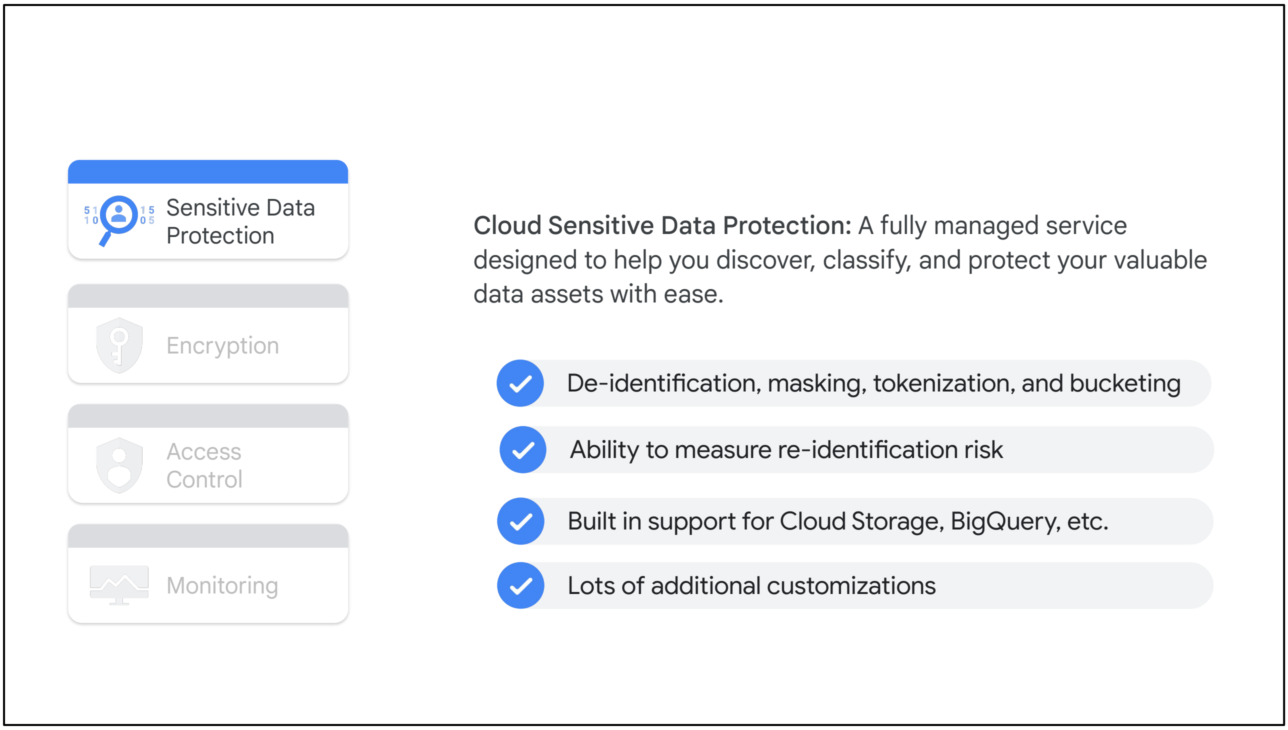
Access Control

Monitoring

Cloud Sensitive Data Protection: A fully managed service designed to help you discover, classify, and protect your valuable data assets with ease.

- ✓ De-identification, masking, tokenization, and bucketing
- ✓ Ability to measure re-identification risk
- ✓ Built in support for Cloud Storage, BigQuery, etc.

- Over 150 built-in information type (or "infoType") detectors and the ability to define your own for automatically scanning for sensitive information. Built-in support for Cloud Storage, BigQuery, BigLake, and Cloud SQL to allow support for additional sources and workloads.



Sensitive Data Protection

Encryption

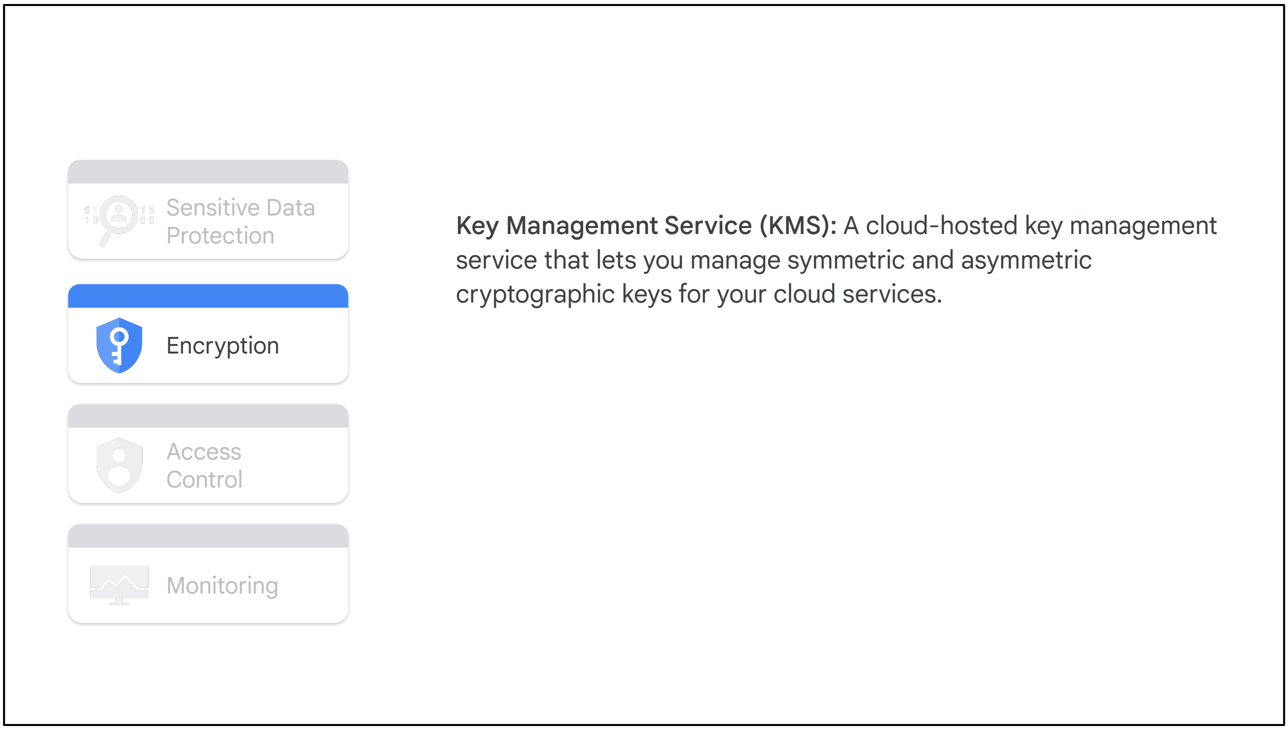
Access Control

Monitoring

Cloud Sensitive Data Protection: A fully managed service designed to help you discover, classify, and protect your valuable data assets with ease.

- ✓ De-identification, masking, tokenization, and bucketing
- ✓ Ability to measure re-identification risk
- ✓ Built in support for Cloud Storage, BigQuery, etc.
- ✓ Lots of additional customizations

- Lots of additional customizations such as adjust detection thresholds, create detection rules, and so on.



Sensitive Data Protection

Encryption

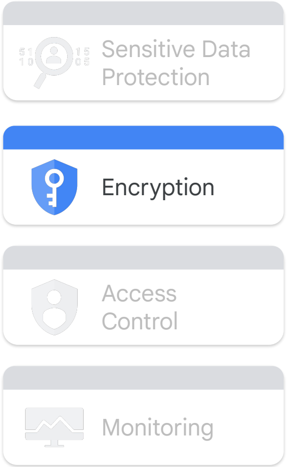
Access Control

Monitoring

Key Management Service (KMS): A cloud-hosted key management service that lets you manage symmetric and asymmetric cryptographic keys for your cloud services.

For encryption, Google provides default encryption at rest, which is fundamental for the privacy of your AI datasets. Data in transit is also encrypted by default using TLS. Google Cloud encrypts your data on the server side before it is written to disk at no additional charge. This data can be previously encrypted if desired.

Add VO: Google Cloud’s Key Management Service is a cloud-hosted service that lets you manage symmetric and asymmetric cryptographic keys for your cloud services.



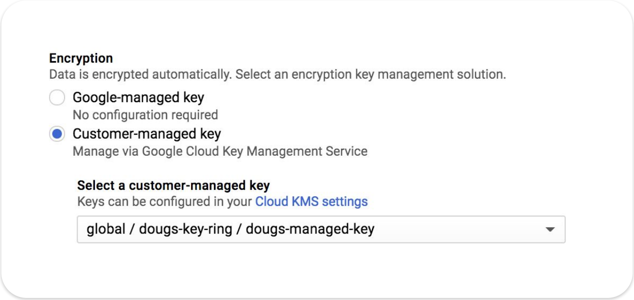
Sensitive Data Protection

Encryption

Access Control

Monitoring

Key Management Service (KMS): A cloud-hosted key management service that lets you manage symmetric and asymmetric cryptographic keys for your cloud services.



Encryption

Data is encrypted automatically. Select an encryption key management solution.

○ Google-managed key
No configuration required

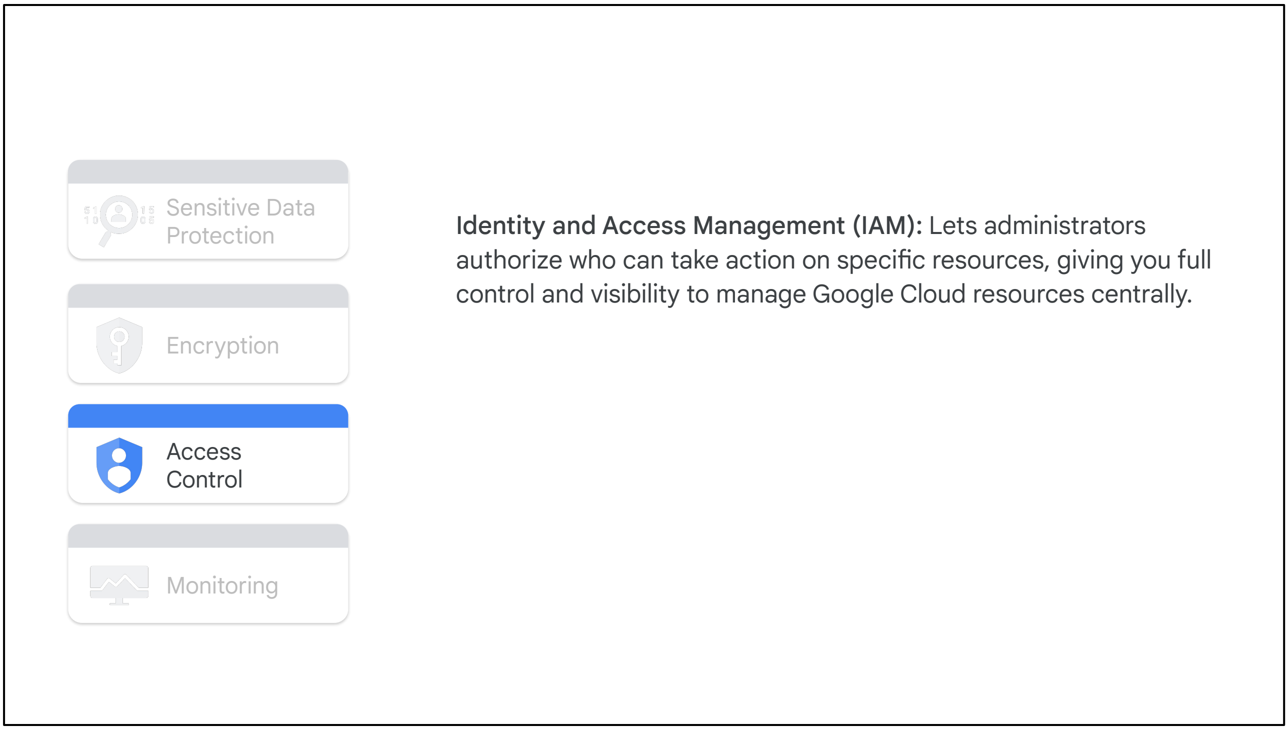
● Customer-managed key
Manage via Google Cloud Key Management Service

Select a customer-managed key

Keys can be configured in your Cloud KMS settings

global / dougs-key-ring / dougs-managed-key ▾

You can use Cloud KMS to extend your control over encryption keys. This allows you to create, import, and manage cryptographic keys, and perform cryptographic operations in a single centralized cloud service.



Sensitive Data Protection

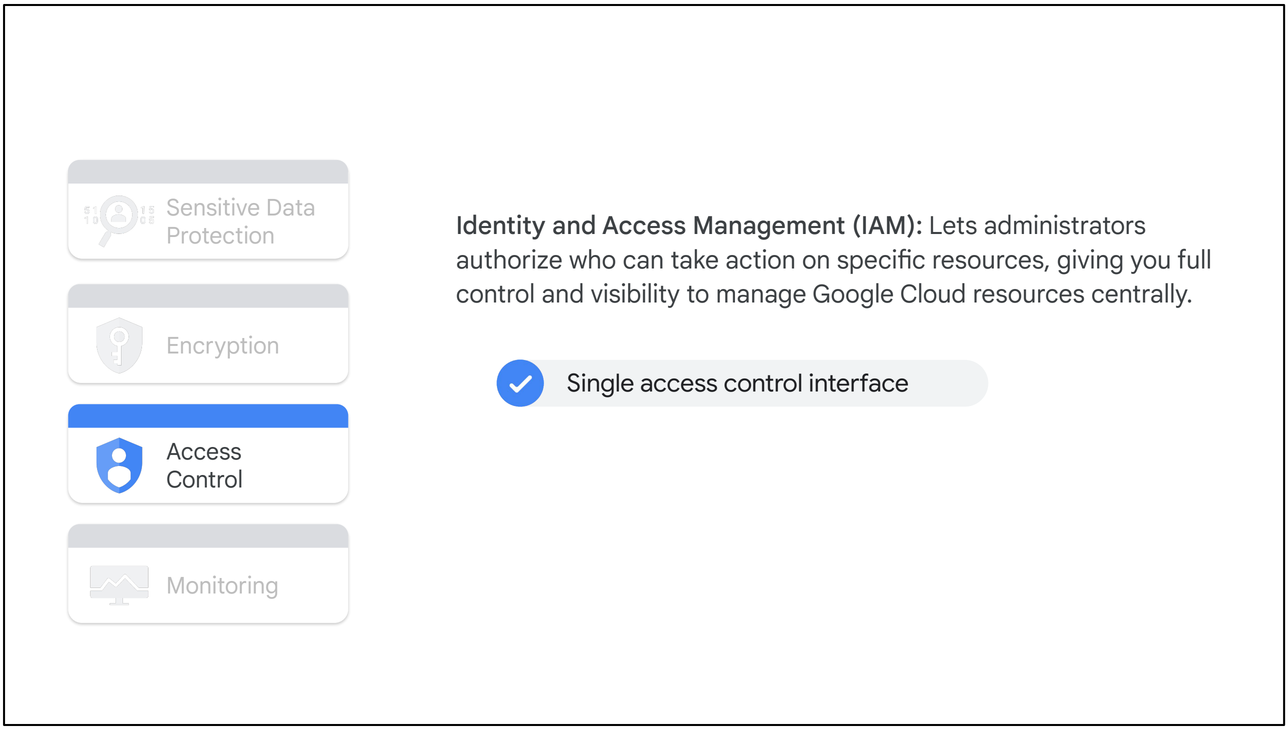
Encryption

Access Control

Monitoring

Identity and Access Management (IAM): Lets administrators authorize who can take action on specific resources, giving you full control and visibility to manage Google Cloud resources centrally.

For access control, Google uses IAM services which give you full control and visibility to manage Google Cloud resources centrally.



Sensitive Data Protection

Encryption

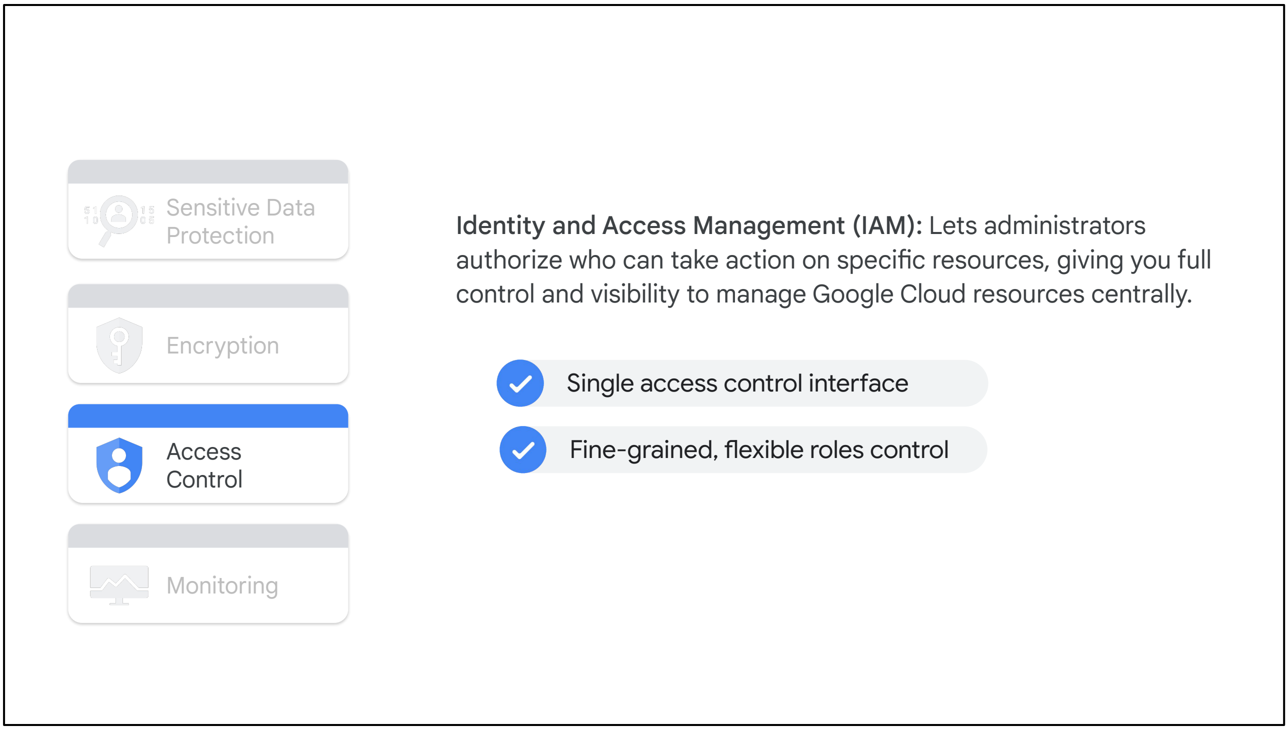
Access Control

Monitoring

Identity and Access Management (IAM): Lets administrators authorize who can take action on specific resources, giving you full control and visibility to manage Google Cloud resources centrally.

✓ Single access control interface

IAM provides all the functionalities you'd expect for access management within a single access interface.

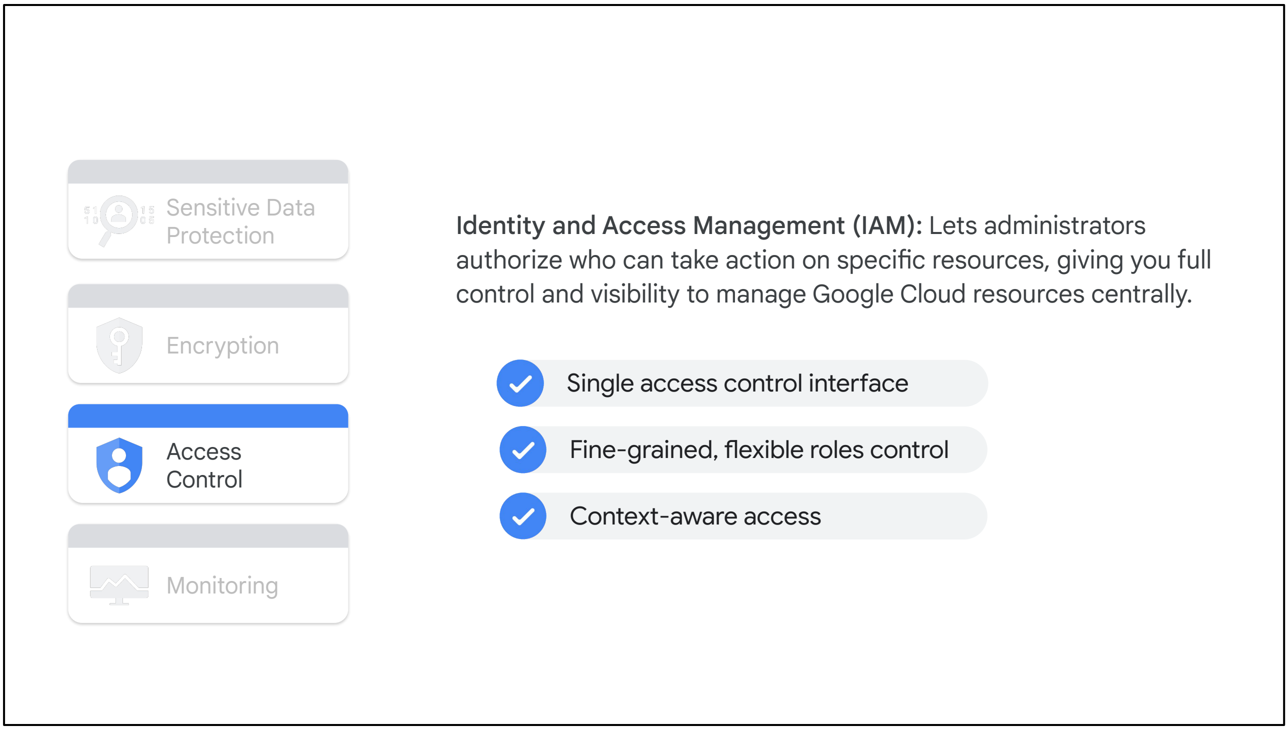


- Sensitive Data Protection
- Encryption
- Access Control
- Monitoring

Identity and Access Management (IAM): Lets administrators authorize who can take action on specific resources, giving you full control and visibility to manage Google Cloud resources centrally.

- ✓ Single access control interface
- ✓ Fine-grained, flexible roles control

such as fine-grained roles, built-in audit trail, and workforce identity federation to use external providers as well.



Sensitive Data Protection

Encryption

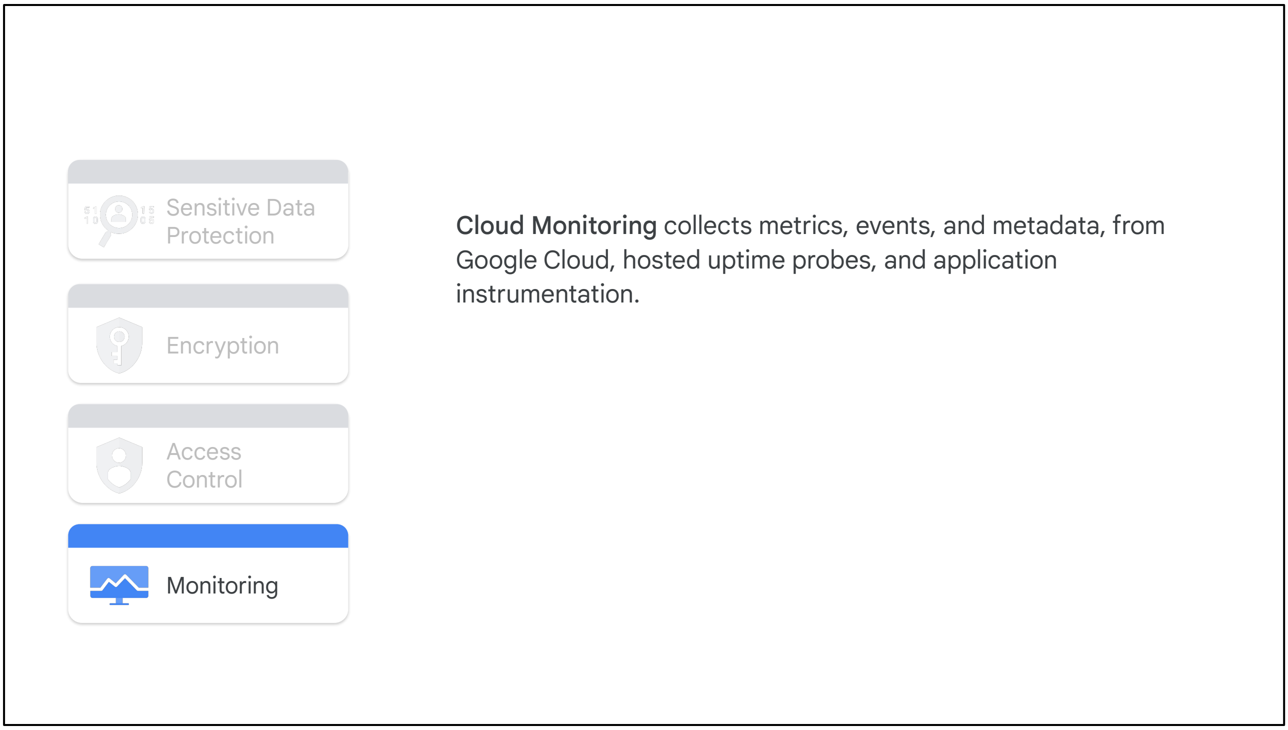
Access Control

Monitoring

Identity and Access Management (IAM): Lets administrators authorize who can take action on specific resources, giving you full control and visibility to manage Google Cloud resources centrally.

- ✓ Single access control interface
- ✓ Fine-grained, flexible roles control
- ✓ Context-aware access

To secure your ML applications, set IAM permissions on the data, models, and serving endpoints following the principle of least privilege.



Sensitive Data Protection

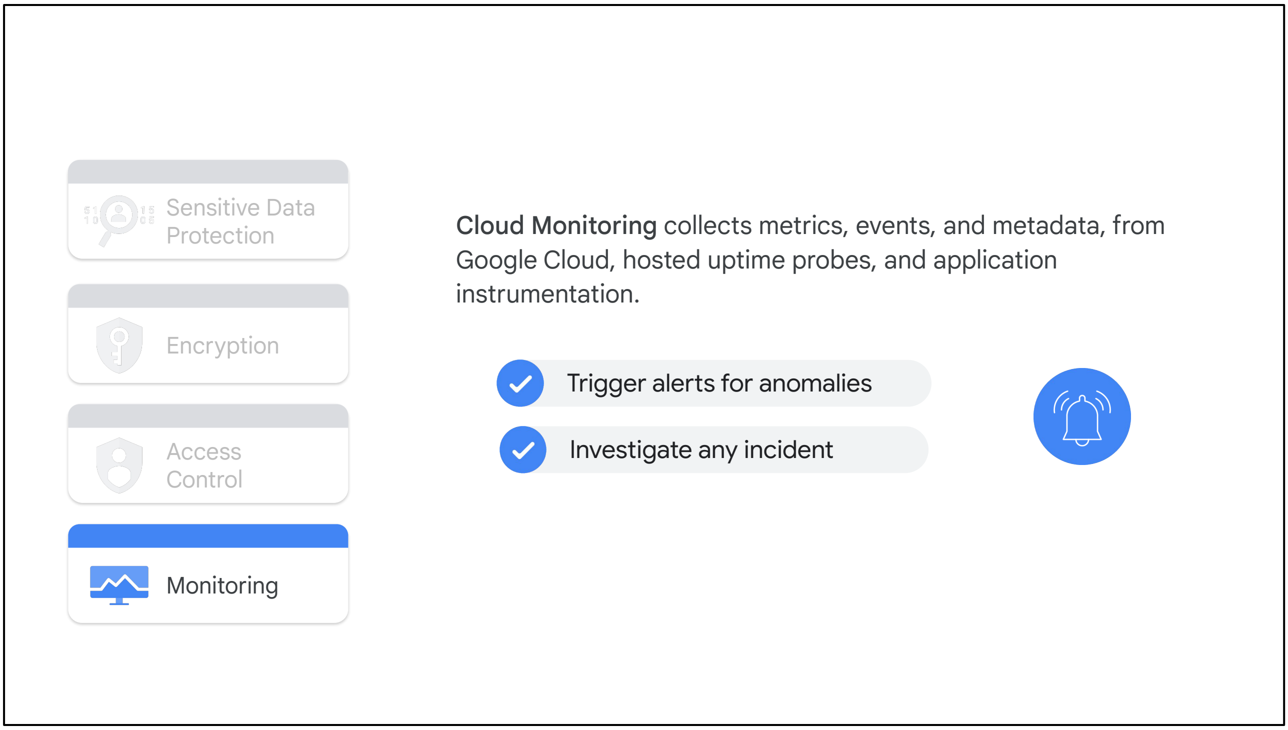
Encryption

Access Control

Monitoring

Cloud Monitoring collects metrics, events, and metadata, from Google Cloud, hosted uptime probes, and application instrumentation.

Cloud Monitoring collects metrics, events, and metadata from Google Cloud, hosted uptime probes, and application instrumentation.



Sensitive Data Protection

Encryption

Access Control

Monitoring

Cloud Monitoring collects metrics, events, and metadata, from Google Cloud, hosted uptime probes, and application instrumentation.

- ✓ Trigger alerts for anomalies
- ✓ Investigate any incident

What's of interest from a security perspective is that every request is logged. This traceability gives you the ability to be notified for anomalies based on custom rules and policies, so that you can investigate any incident.

01 Overview of AI Privacy

02 Privacy in Training Data: De-identification techniques

03 Privacy in Training Data: Randomization techniques

04 Privacy in Machine Learning Training: DP-SGD

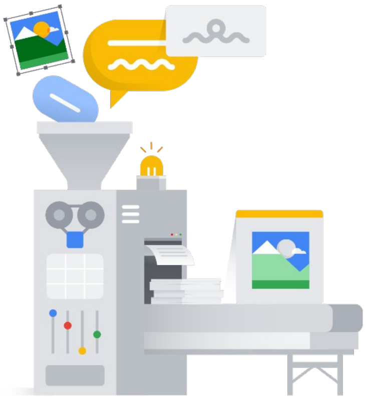
05 Privacy in Machine Learning Training: Federated Learning

06 System Security on Google Cloud

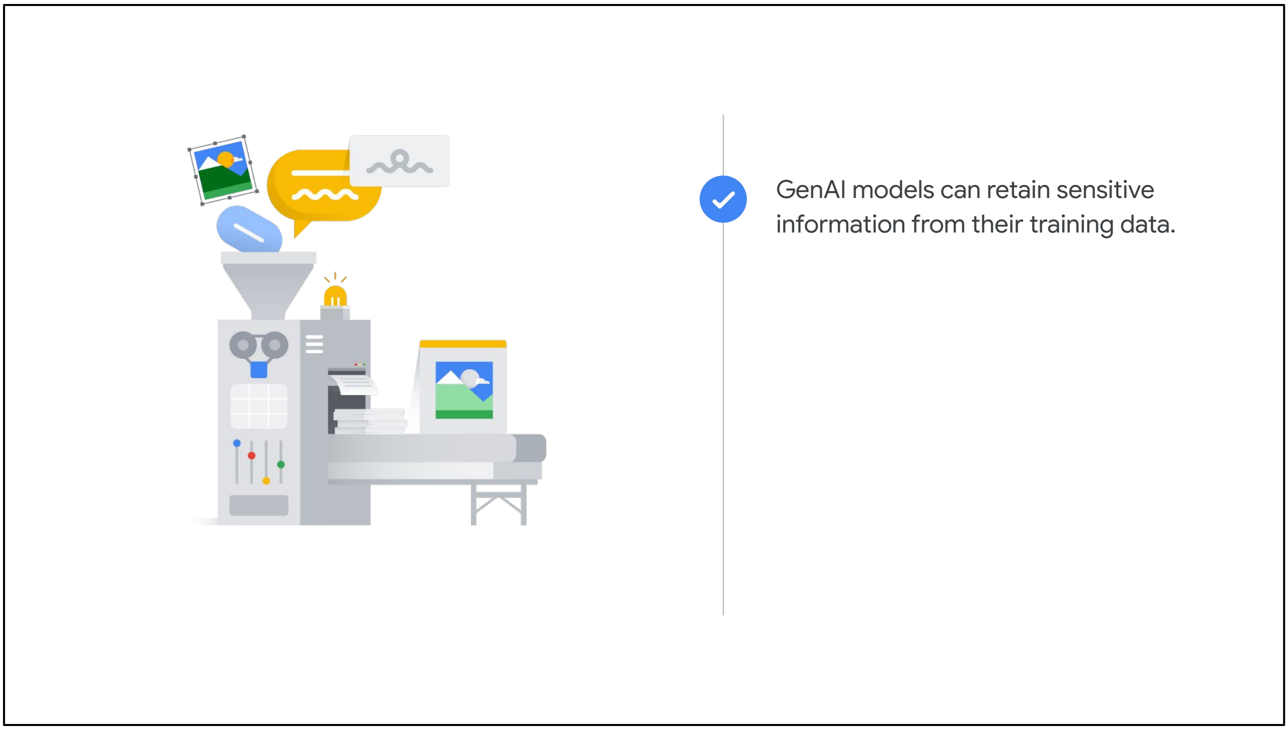
07 System Security and Generative AI

08 Lab: Differential Privacy in Machine Learning with TensorFlow Privacy

Let’s look at system security and how it relates to generative AI.

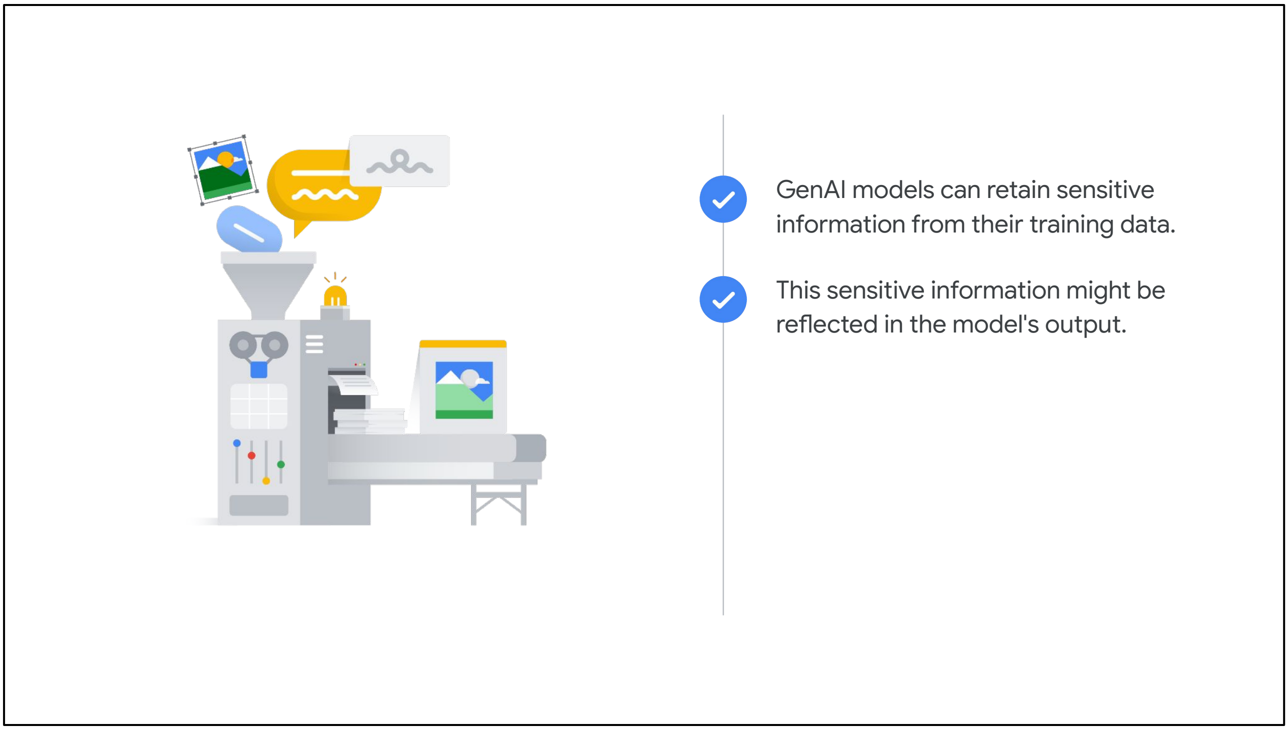


Generative AI models are trained on vast sources of unstructured data that pull from a range of sources.



✓ GenAI models can retain sensitive information from their training data.

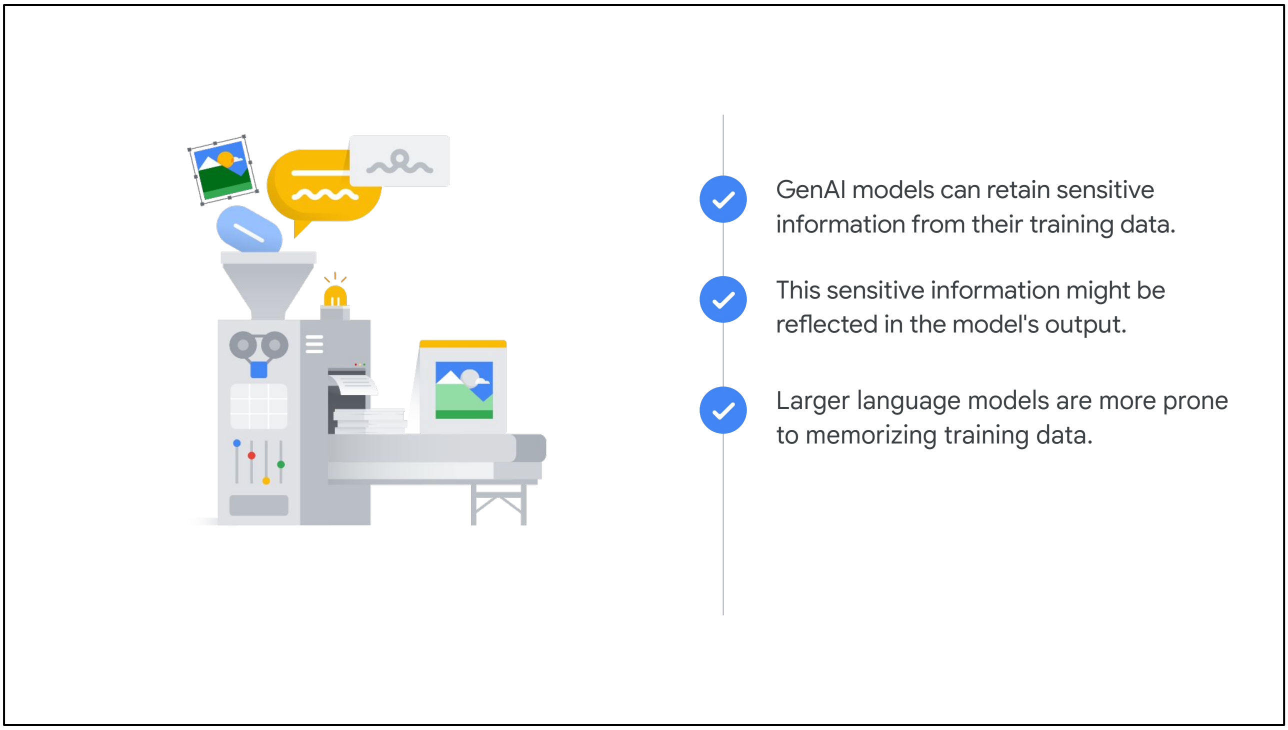
It is likely that they could contain some sensitive data,



✓ GenAI models can retain sensitive information from their training data.

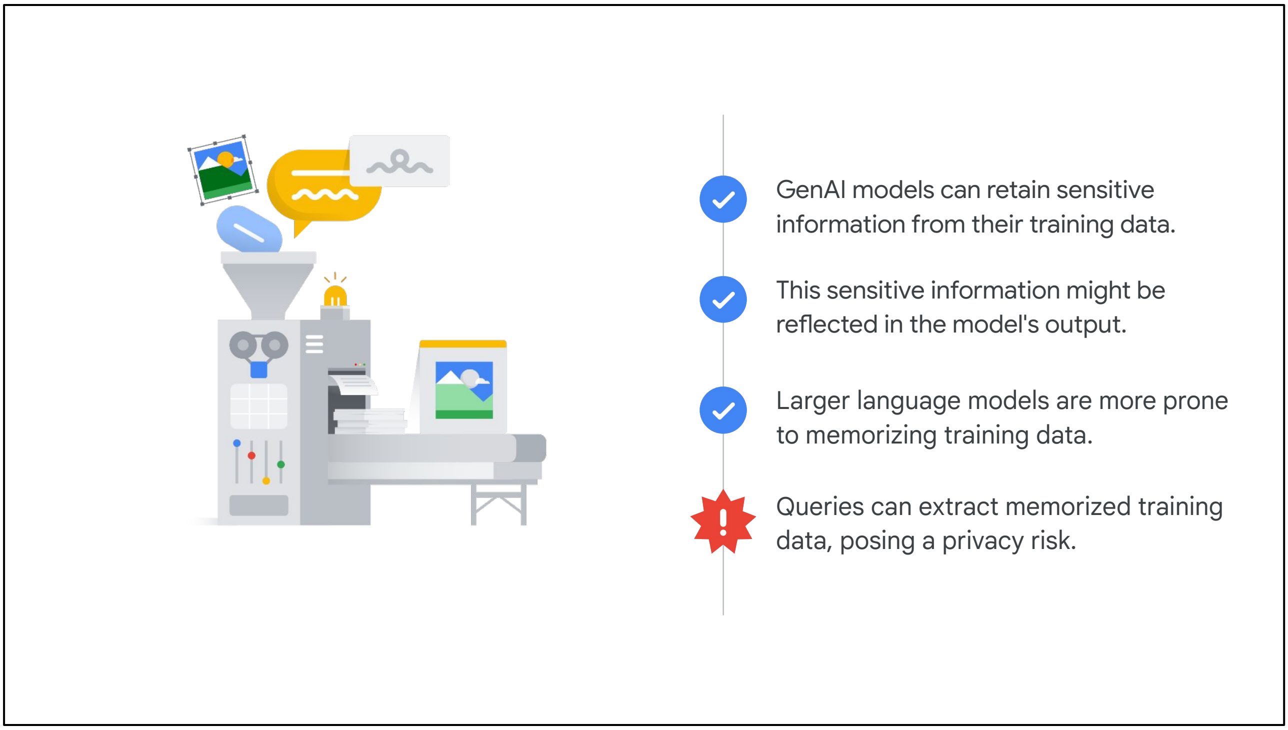
✓ This sensitive information might be reflected in the model's output.

and it is possible that they could reflect some of these private details in the output



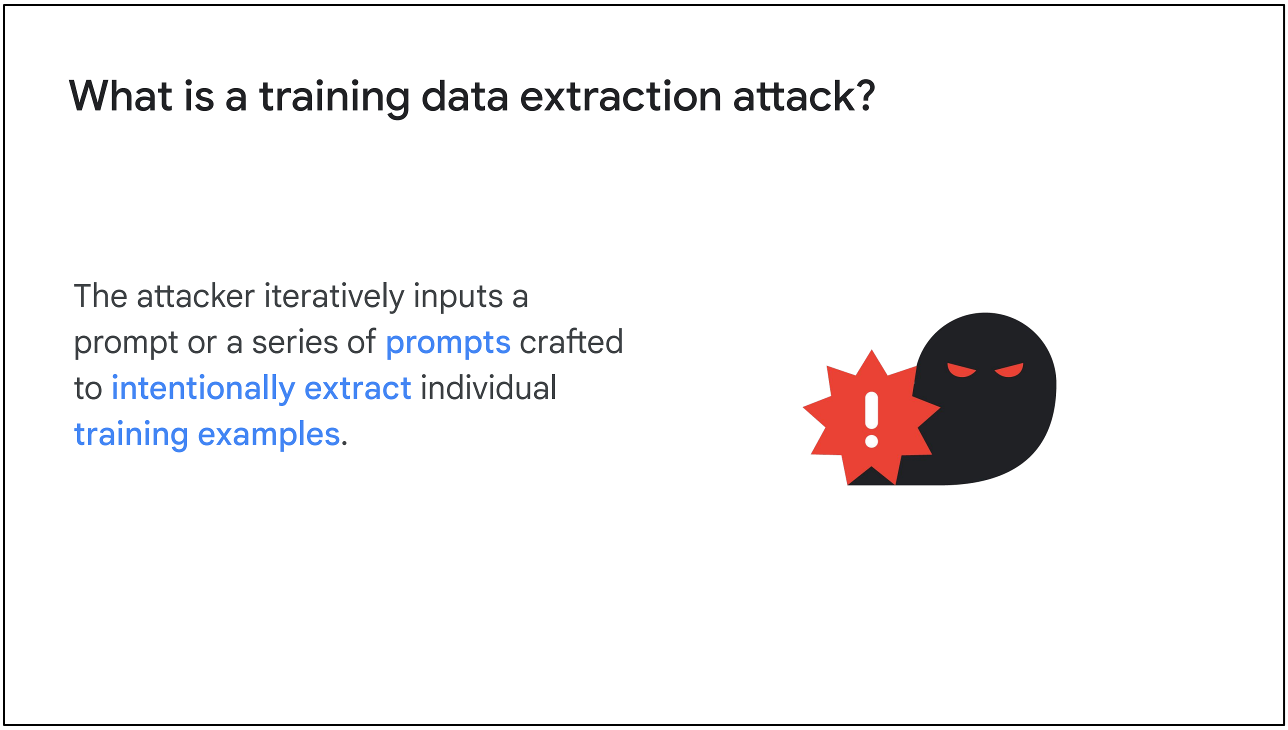
- ✓ GenAI models can retain sensitive information from their training data.
- ✓ This sensitive information might be reflected in the model's output.
- ✓ Larger language models are more prone to memorizing training data.

In fact, the larger the language model, the more easily it memorizes training data.
It is a known fact that, through a simple query,



- ✓ GenAI models can retain sensitive information from their training data.
- ✓ This sensitive information might be reflected in the model's output.
- ✓ Larger language models are more prone to memorizing training data.
- ! Queries can extract memorized training data, posing a privacy risk.

it is not impossible to extract specific pieces of training data that the model has memorized. This makes generative AI models particularly sensitive to training data extraction attacks.

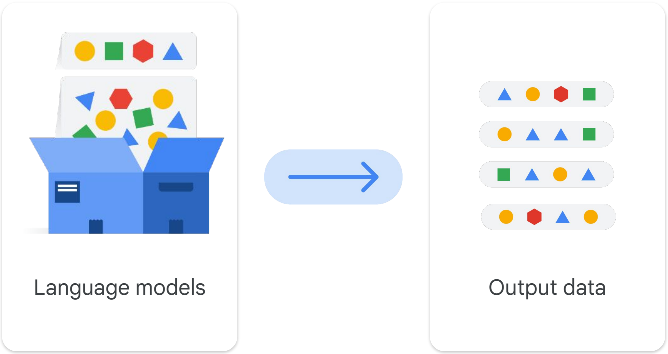


What is a training data extraction attack?

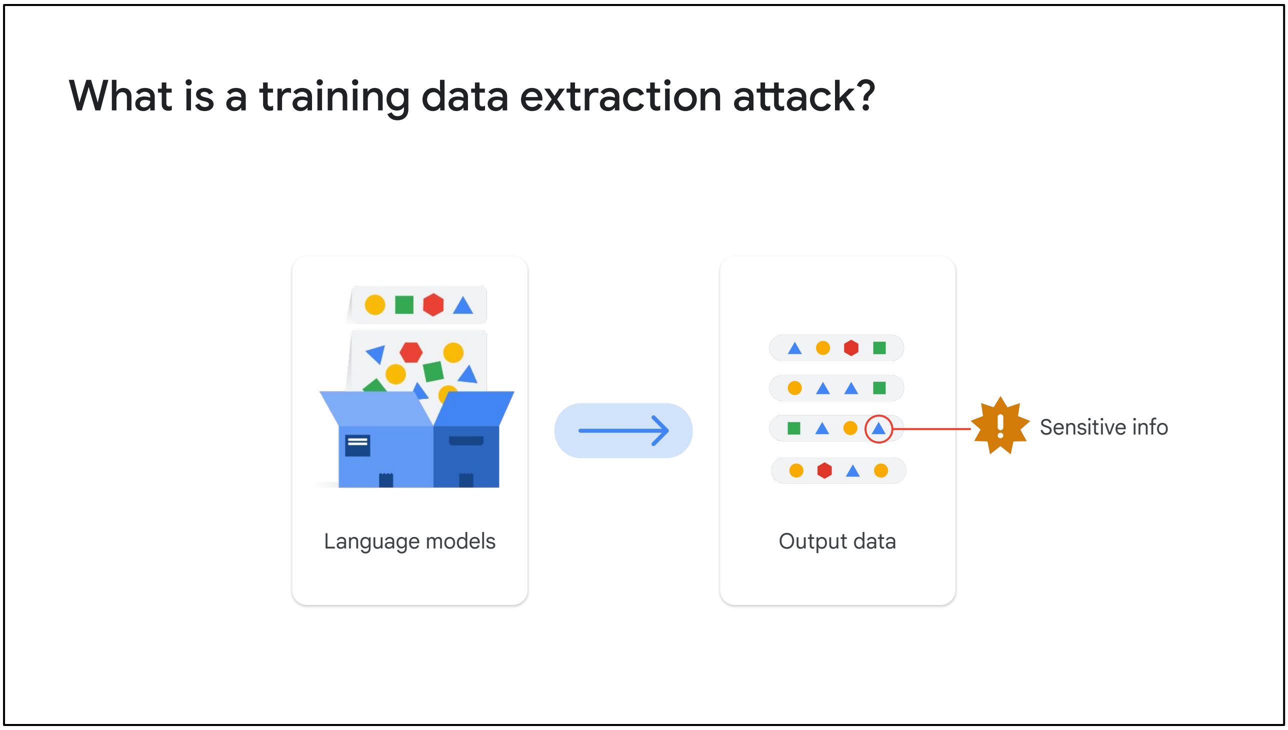
The attacker iteratively inputs a prompt or a series of prompts crafted to intentionally extract individual training examples.

In a training data extraction attack, the attacker iteratively inputs a prompt or a series of prompts crafted to intentionally extract individual model training examples.
This type of attack has the greatest potential for harm when applied to a model that is available to the public, but the dataset used in the training is not publicly available.

What is a training data extraction attack?



By design, language models make it very easy to generate a large amount of output data. By seeding the model with random short phrases, the model can generate millions of continuations, for example probable phrases that complete a sentence. Most of the time, these continuations are benign strings of sensible text.



What is a training data extraction attack?

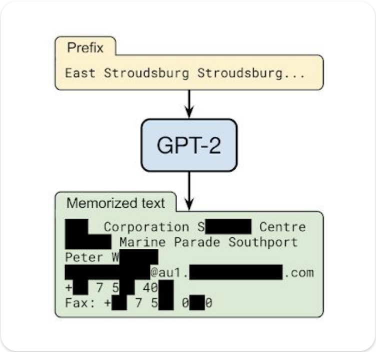
Language models

Output data

Sensitive info

Sometimes, they can leak sensitive information.

What is a training data extraction attack?



Prefix

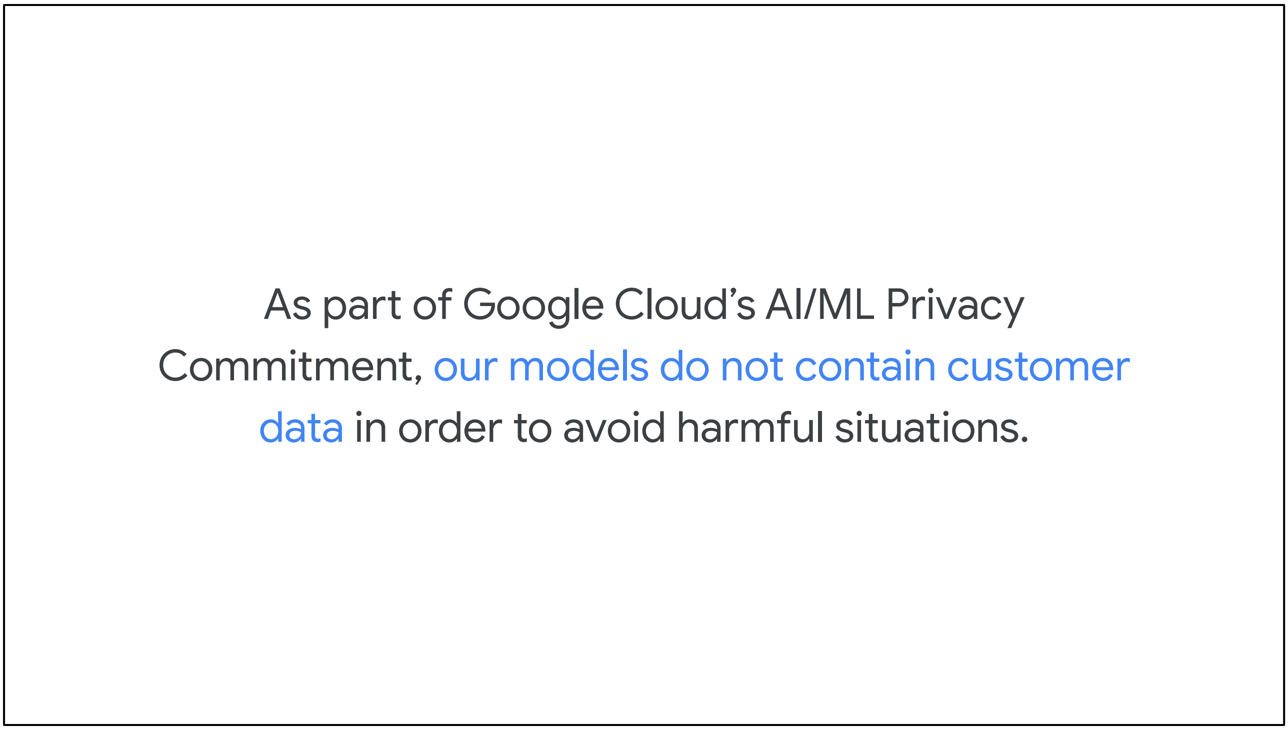
```
East Stroudsburg Stroudsburg...
```

GPT-2

Memorized text

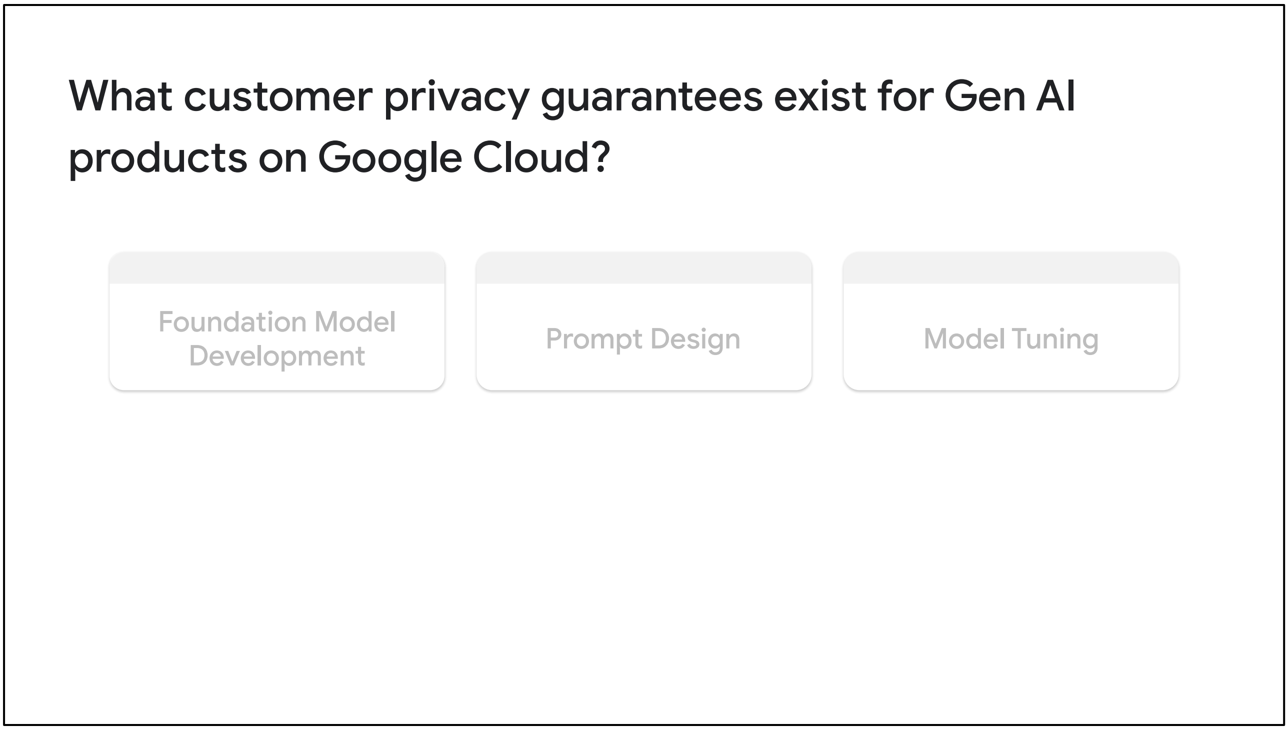
```
[redacted] Corporation S[redacted] Centre
[redacted] Marine Parade Southport
Peter W[redacted]
[redacted]@au1.[redacted].com
+[redacted] 7 5[redacted] 40[redacted]
Fax: +[redacted] 7 5[redacted] 0[redacted]0
```

Let’s look at one example taken from one of the first papers to analyze this adversarial attack. A user enters the prompt “East Stroudsburg Stroudsburg...” into the GPT-2 language model. The model then autocompletes a long block of text that contains the full name, phone number, email address, and physical address of a particular person whose information happened to be included in GPT-2’s training data. This results in the model allowing the user to obtain sensitive information.



As part of Google Cloud’s AI/ML Privacy Commitment, our models do not contain customer data in order to avoid harmful situations.

At Google, data governance is a key aspect of privacy. As part of Google Cloud’s AI/ML Privacy Commitment, our models do not contain customer data in order to avoid situations such as this.



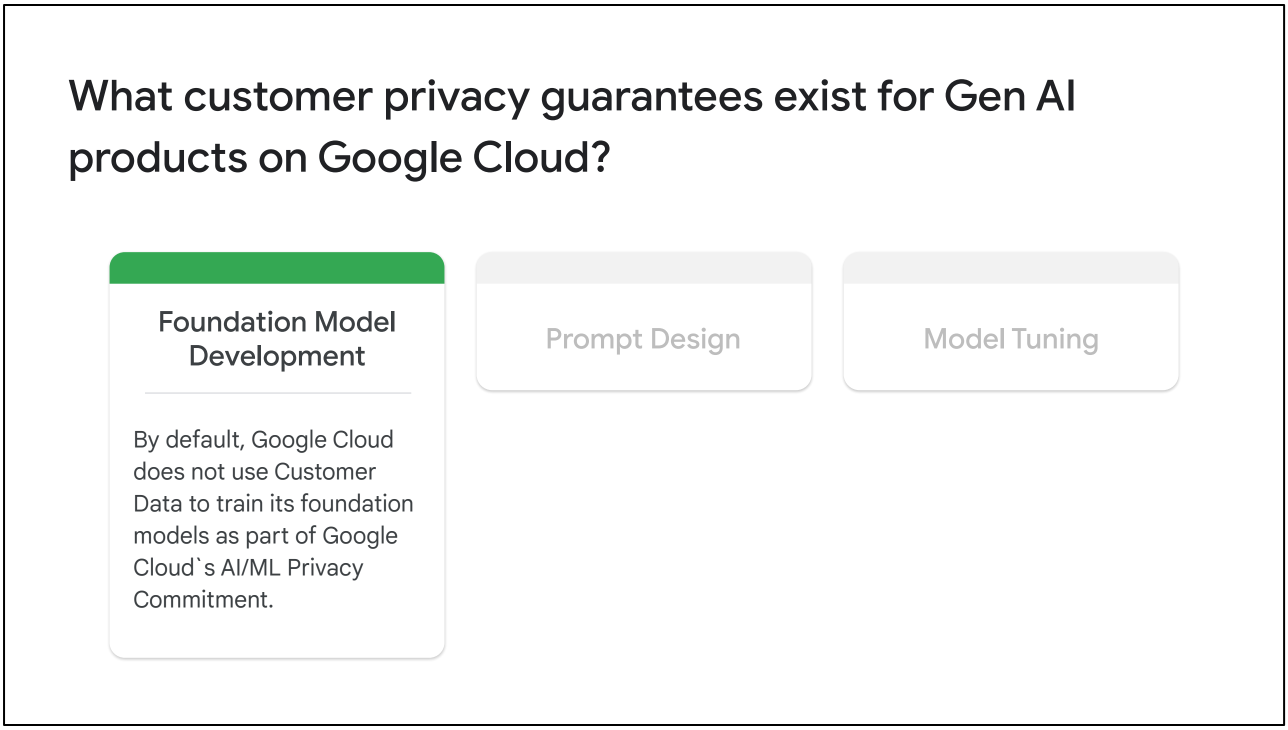
What customer privacy guarantees exist for Gen AI products on Google Cloud?

Foundation Model Development

Prompt Design

Model Tuning

Google’s commitment for privacy extends to Google Cloud generative AI products and solutions. Google Cloud guarantees the following:



What customer privacy guarantees exist for Gen AI products on Google Cloud?

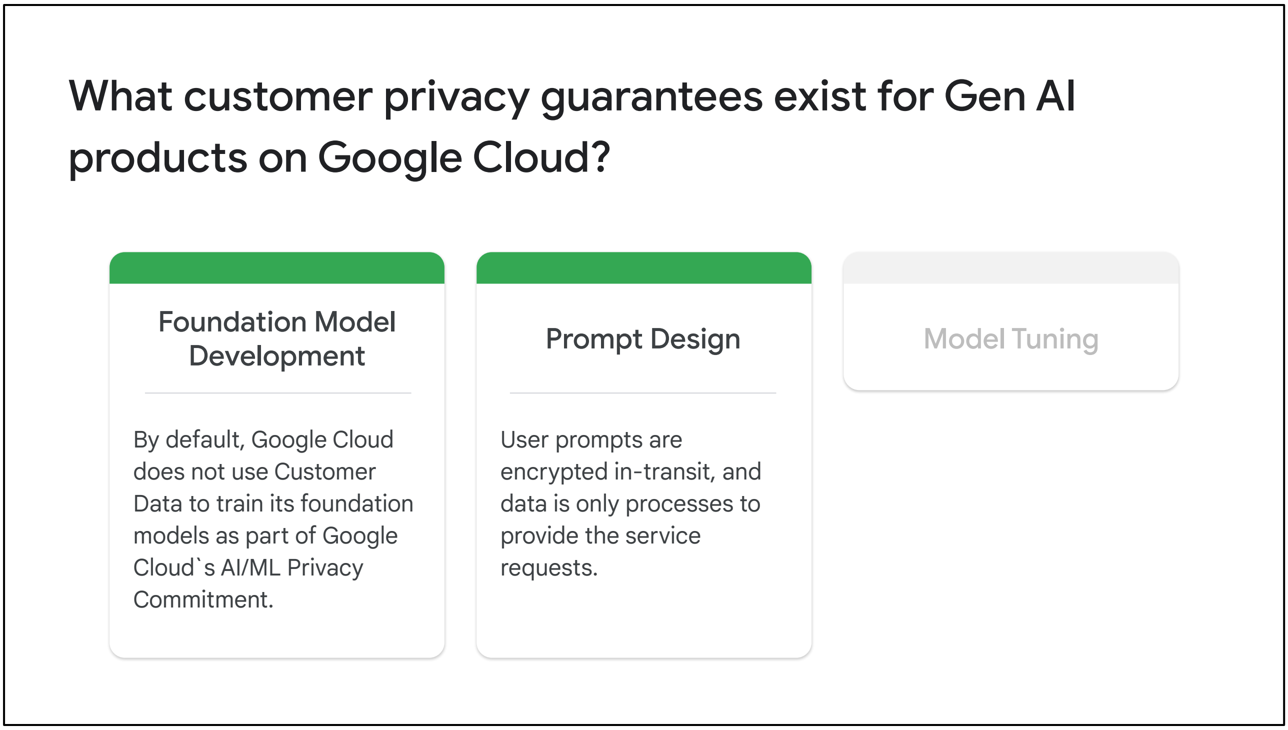
Foundation Model Development

By default, Google Cloud does not use Customer Data to train its foundation models as part of Google Cloud`s AI/ML Privacy Commitment.

Prompt Design

Model Tuning

1. The foundation model development is privacy compliant. By default, Google Cloud does not use Customer Data to train its foundation models as part of Google Cloud`s AI/ML Privacy Commitment.



What customer privacy guarantees exist for Gen AI products on Google Cloud?

Foundation Model Development

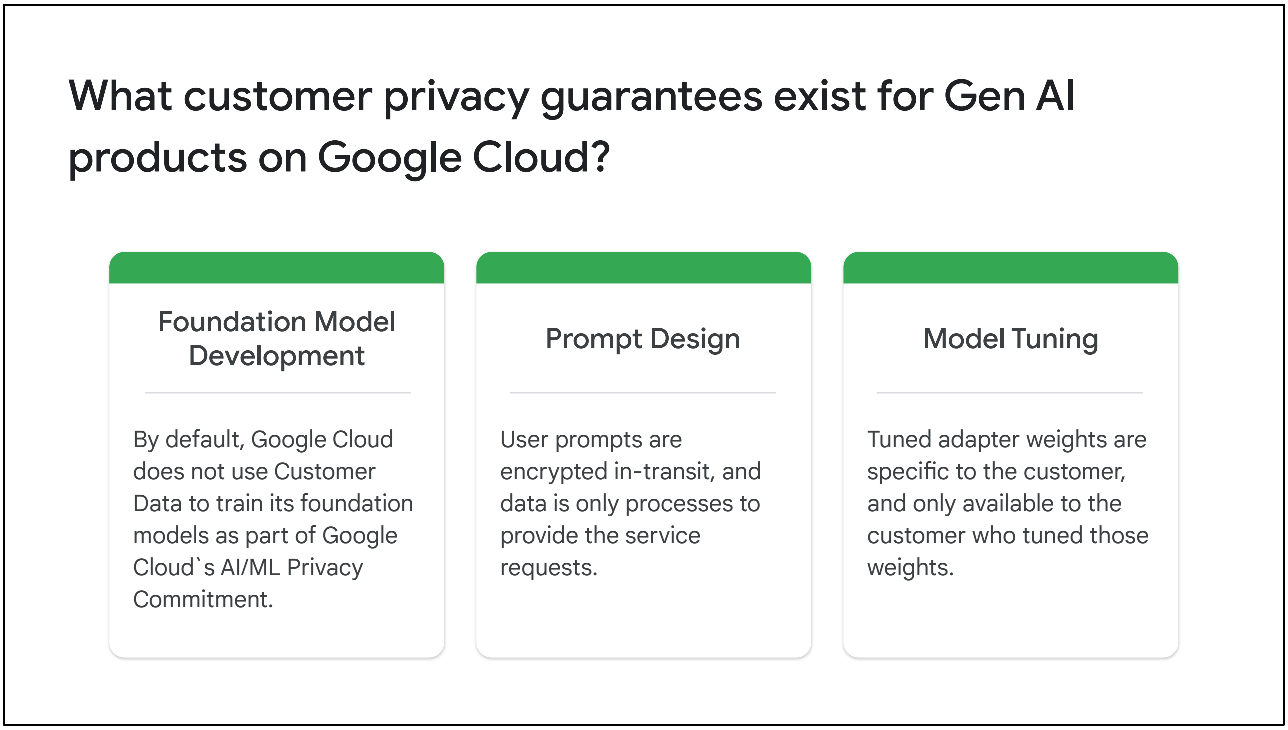
By default, Google Cloud does not use Customer Data to train its foundation models as part of Google Cloud`s AI/ML Privacy Commitment.

Prompt Design

User prompts are encrypted in-transit, and data is only processes to provide the service requests.

Model Tuning

2. When a request is submitted with a prompt to our foundation model, Customer Data is encrypted in-transit and input to the foundation model to generate a response. Google processes Customer Data to provide the service requested only.



What customer privacy guarantees exist for Gen AI products on Google Cloud?

Foundation Model Development

By default, Google Cloud does not use Customer Data to train its foundation models as part of Google Cloud`s AI/ML Privacy Commitment.

Prompt Design

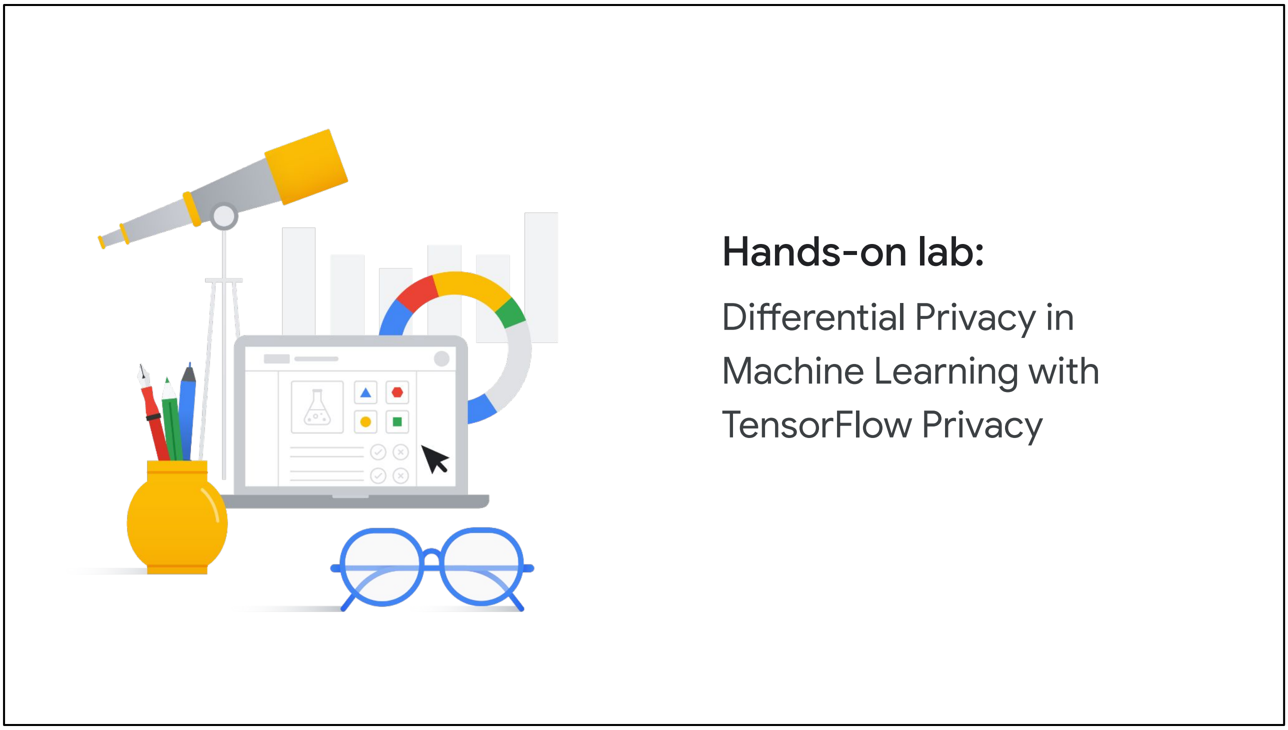
User prompts are encrypted in-transit, and data is only processes to provide the service requests.

Model Tuning

Tuned adapter weights are specific to the customer, and only available to the customer who tuned those weights.

3. The adapter weights are specific to the customer, and only available to the customer who tuned those weights. During inference, the foundation model receives the adapter weights, runs through the request and returns the results, without modifying the foundation model or storing the request. Input data is Customer Data and is stored securely at every step along the way—encrypted at rest and in transit. Tuned weights are also stored securely, and Customers will have sole access to use any tuned models. The customer is able to control the encryption of stored adapters by using customer-managed encryption keys (CMEK), and can delete adapter weights at any time.

And that is how Google Cloud helps ensure security best practices are applied to privacy when training machine learning models.



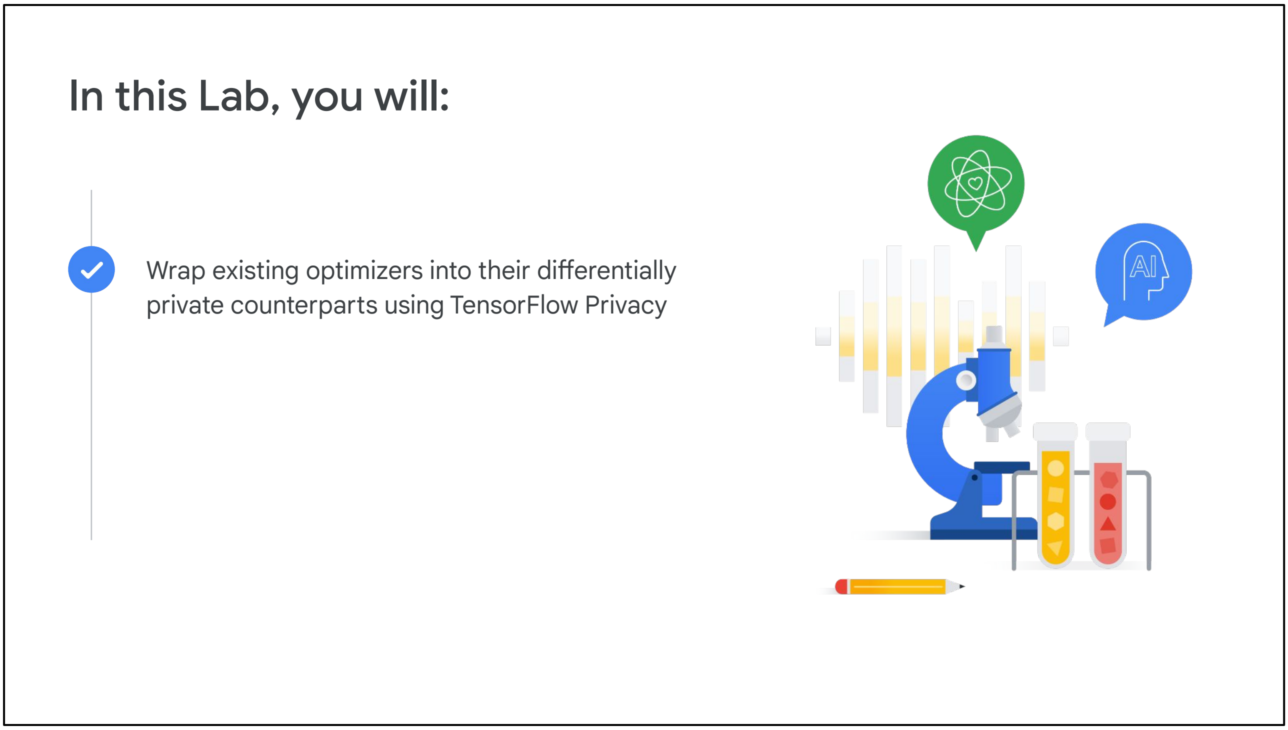
Hands-on lab:

Differential Privacy in Machine Learning with TensorFlow Privacy

Now let’s perform an exercise in a hands-on lab.

In this Lab, you will:

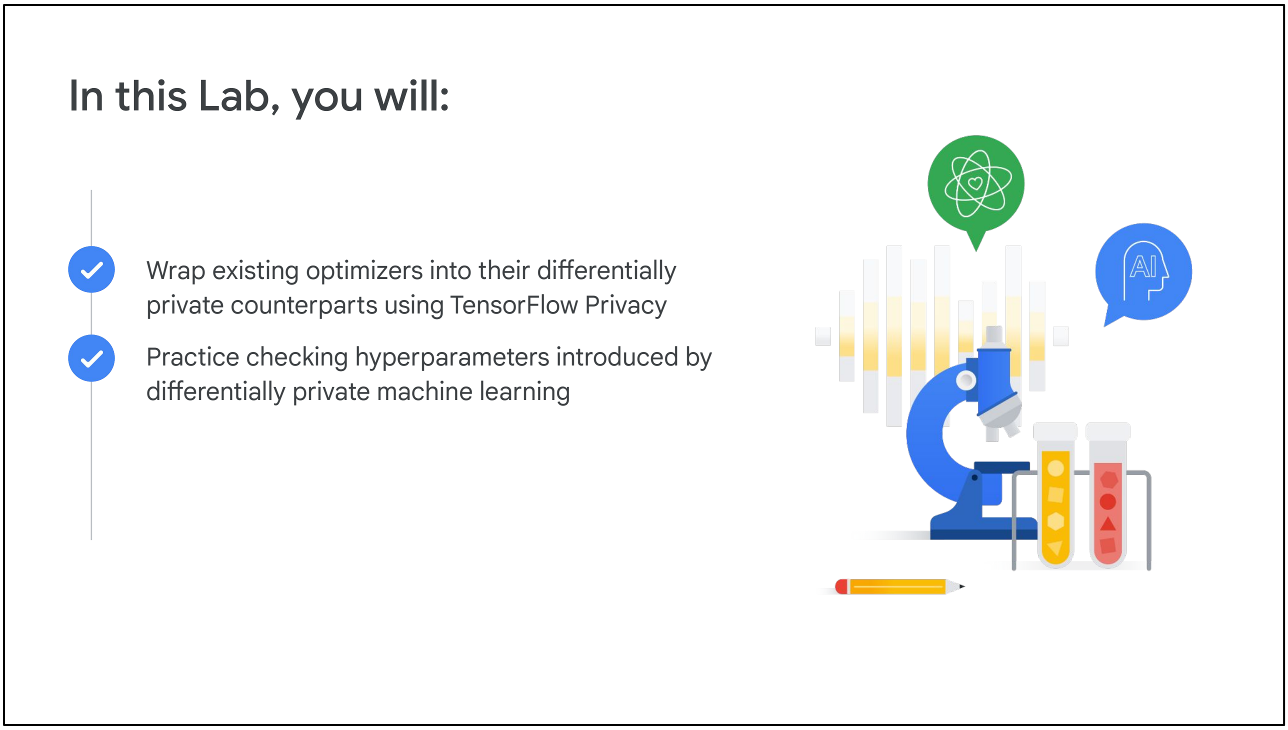
This lab helps you learn how to use differential privacy in machine learning using TensorFlow Privacy. In this lab, you will:



In this Lab, you will:

✓ Wrap existing optimizers into their differentially private counterparts using TensorFlow Privacy

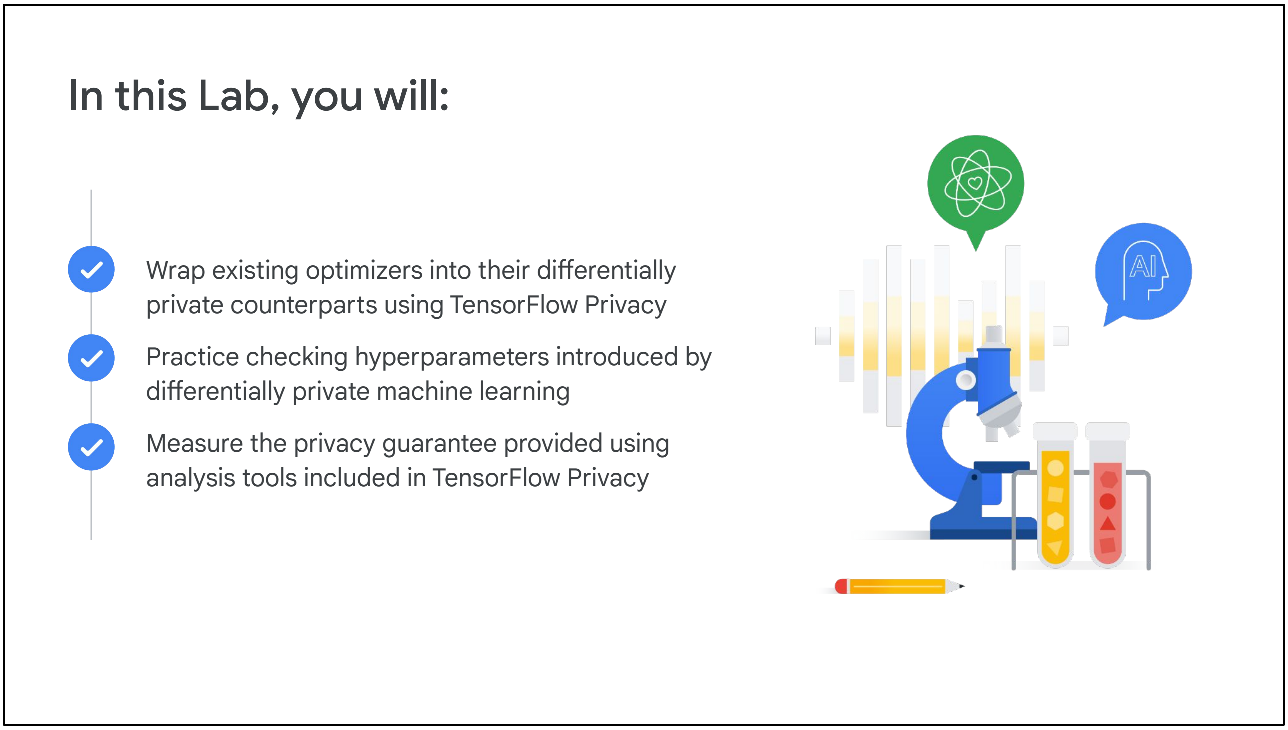
- Wrap existing optimizers into their differentially private counterparts using TensorFlow Privacy.



In this Lab, you will:

- ✓ Wrap existing optimizers into their differentially private counterparts using TensorFlow Privacy
- ✓ Practice checking hyperparameters introduced by differentially private machine learning

- Practice checking hyperparameters introduced by differentially private machine learning.



In this Lab, you will:

- ✓ Wrap existing optimizers into their differentially private counterparts using TensorFlow Privacy
- ✓ Practice checking hyperparameters introduced by differentially private machine learning
- ✓ Measure the privacy guarantee provided using analysis tools included in TensorFlow Privacy

- Measure the privacy guarantee provided using analysis tools included in TensorFlow Privacy.