Teaching Aide: Privacy Preserving Cryptography

By Azeezat Lawal, Dami Ogunnupebi, Pradip Ghimire, Abdul Osuwa and Tarun Sidhu

Introduction/Abstract

Privacy Preserving Cryptography is the cryptographic method that allows data to be shared and processed without revealing sensitive information to other unauthorized parties. **For the purpose of this teaching aide, we will refer to Privacy Preserving Cryptography as PPC.** The general concept of PPC lies in being able to use data or information without full, unauthorized access to the original data. In order to achieve this, PPC employs a number of techniques that share its goal, but reach this goal in different ways. For example, while Secure Multi-Party Computation uses a form of obfuscation to use data (i.e. either only bits of the original data is shared with a group, or the data itself is encrypted before use), Zero Knowledge Proofs subtly hinges on the idea that proof can be sent on an individual without fully revealing the details of that person (e.g. the process of credit-checking a person for loan application reviews).

While we explore these techniques a bit further, the only similarity you will notice is the constant notion of being able to safely use data for various purposes without having access to its original form. PPC is a cryptographic application that was built with the very essence of privacy in mind, and with its different techniques that can be applied for different fitting scenarios, PPC continues to fulfil that goal. With PPC, companies can collaborate securely, while keeping privacy in mind, and depending on their industry, they can adopt the most appropriate cryptographic techniques it provides.

Our group's Teaching Aide will explore the different techniques used in Privacy Preserving Cryptography, some of its applications and a few challenges to its application. Overall, it's not a fully widely adopted concept (at least not yet), but it is forecasted by Gartner to be fully adopted in about 5-10 years (Lowans, 2020).

Privacy-Preserving Cryptography and Its Techniques

As indicated above, when you review the techniques that make up PPC, you will see that they are very similar in that, at their very core and foundation, they are made up of the one goal that PPC aims to achieve: to preserve privacy. We will now go into a few details about how these techniques work; some cover very technical details while some do not; and a few examples of how these techniques work theoretically & practically.

Zero-Knowledge Proofs

Imagine you need to prove to someone that you know a secret, like a password or the solution to a puzzle, without revealing the key/secret. Normally, proving what you know means giving away at least part of the secret. Zero-knowledge proofs (ZKPs) solve this problem. They are an excellent cryptographic method to convincingly show something is true while revealing no new information. That is, you can effectively show someone you have knowledge or access to something without showing the "how" or the "what." Think of it as proving you have the key to a lock without ever handing over the key or even letting them see it. (Source)

ZKPs are among the most powerful privacy tools in today's digital security and are utilized in protocols such as blockchain, secure electronic voting, and passwordless authentication. The idea first appeared in 1985 and has since grown into a critical field of research and application. From protecting financial transactions to securing identities, zero-knowledge proofs allow verification and trust in a world where sharing sensitive data can be risky. This article explains ZKPs, their history, operation, and why they matter today.

A short history of ZKPs

Zero-knowledge in cryptography first appeared in the 1985 paper "**The knowledge complexity of interactive proof systems [GMR85]**" by pioneers Shafi Goldwasser, Silvio Micali, and Charles Rackoff. They provide a definition of zero-knowledge proofs that is widely used today.

"A zero-knowledge protocol is a technique by which one party (the prover) can demonstrate to another party (the verifier) that a particular statement is true without disclosing any additional information besides that fact." To put it some other way, "I can prove to you that this statement involving X is true, but I won't tell you that I know X."

Zero-knowledge proofs must satisfy three properties:

- **Completeness:** if the statement is true, an honest verifier will be convinced by an honest prover.
- **Soundness:** if the statement is false, no dishonest prover can convince the honest verifier. The proof systems are truthful and do not allow cheating.
- **Zero-Knowledge:** if the statement is true, no verifier learns anything other than the fact that the statement is true

Real-World Applications of Zero-Knowledge Proofs

Zero-knowledge proofs are used in actual systems today and are not merely theoretical:

- **Blockchain privacy:** ZKPs enable systems like Zcash to enable "imagine sending money where everyone can see the transaction happened but nobody can see who sent how much."
- **Electronic voting:** "Voters can demonstrate that they cast a ballot without disclosing their selection, guaranteeing free and confidential elections."
- **Passwordless authentication:** "Verify user password without ever sending the password."
- **Access Control & Gaming:** Check a player's credentials or accomplishments without exposing sensitive information.

Types of Zero-Knowledge Proofs:

Interactive Zero-Knowledge Proofs

Initially introduced by Goldwasser, Micali, and Rackoff (1985), interactive ZKPs are between a verifier and a prover who communicate with multiple messages. The verifier presents random challenges, and the prover responds in an effort to convince the verifier without revealing the secret. Because the verifier makes non-predicable choices, an unethical prover cannot invariably forge proofs. These are exemplified in graph isomorphism protocols and proofs of discrete logarithms. Interactive ZKPs are the foundation of cryptography but are less practical in today's systems of decentralization because a live connection and multiple rounds of communication are required. They are, however, still crucial in the comprehension of ZKP theory and are actually applied in secure voting and authentication.

Non-Interactive Zero-Knowledge Proofs (NIZKs)

Recommended by Blum, Feldman, and Micali (1988), NIZKs do away with the need for interactive communication. The prover generates one proof with common reference string (CRS) or public randomness. Anyone may verify the proof offline. NIZKs are highly popular with regard to blockchain privacy protocols, digital signature, and secret asset transfer. They are superior in practice with respect to interactive ZKPs, especially with decentralised networks. They, however, normally require a trusted setup in the generation of the CRS—if it gets compromised, the entirety of the system's security may be lost. Despite the downside, NIZKs are what contemporary proof systems including zk-SNARKs, zk-STARKs, and Bulletproofs are founded upon.

zk-SNARKs (Succinct Non-Interactive Arguments of Knowledge)

zk-SNARKs are sophisticated NIZKs with very small proofs that are fast to verify. It was born out of research by Groth (2016) and popularised with Zcash. Elliptic curve pairings and polynomial commitments are used in zk-SNARKs to attest statements with very minimal computational expense. They have the primary advantage of efficiency—verification is in milliseconds, and as such, are ideal for blockchains and protecting transactions with privacy. The primary disadvantage is the need for a trusted setup, one that is a security vulnerability if not performed with care. Variances such as Groth16 and PLONK permit wider universality and malleability. zk-SNARKs are core in scaleable decentralised finance (DeFi) and layer-2 scaleables.

zk-STARKs (Scalable Transparent Arguments of Knowledge)

zk-STARKs were introduced by Eli Ben-Sasson et al. in 2018 to address SNARK limitations. Unlike SNARKs, STARKs are transparent, meaning no trusted setup is required—they rely solely on public randomness. They also use hash functions and information-theoretic security, making them quantum-resistant. STARKs are highly scalable, handling massive computations efficiently, though their proofs are larger than SNARKs. They are ideal for blockchain rollups, verifiable computations, and scalable privacy systems. Projects like StarkWare use zk-STARKs to build high-performance Ethereum layer-2 networks, proving they're a powerful alternative when trust minimization and post-quantum security are priorities.

Bulletproofs

Bulletproofs are short, efficient NIZKs developed by Bünz et al. (2018) and are specifically designed for range proofs—that is, proving a secret number is in a range without revealing it. They do not require a trusted setup and are smaller in footprint than earlier range-proof methods. Bulletproofs use inner-product arguments for brevity, but verification time is proportional to proof size, so for large computations, proof verification is slower than with SNARKs. They are used extensively throughout privacy-oriented cryptocurrencies like Monero for hiding transaction values and in decentralized exchanges for anonymous order books. Bulletproofs are valued for being easy, trustless, and functional over established cryptosystems.

Sigma Protocols

Sigma protocols are three-move interactive ZKPs: commitment, challenge, and response. They are building blocks for many other ZKP systems. A classic example is the Schnorr identification protocol for showing knowledge of a discrete logarithm. Sigma protocols are relatively simple, efficient, and proven secure with usual cryptographic assumptions. They are, however, interactive and less versatile and less compact than SNARKs or STARKs. Due to their modularity, they are an indispensable tool for devising complex proofs, including Fiat–Shamir transformations that turn them into NIZKs. Sigma protocols remain the cornerstone of authentication, secure key exchange, and crypto-protocol design.

PLONK

Gabizon, Williamson, and Ciobotaru introduced PLONK in 2019 as a modern version of zk-SNARK. It employs a universal and updatable trusted setup, allowing for the use of one setup for multiple circuits and the safe addition of randomness by multiple users. To make proof generation and verification more efficient, PLONK uses polynomial commitments and permutation arguments. Compared to earlier SNARKs such as Groth16, it is faster, more flexible, and easier for developers to use. PLONK has also resulted in improvements such as TurboPLONK and UltraPLONK, which are now widely used in Ethereum layer-2 scaling, DeFi protocols, and cross-chain bridges for secure and private computation.

Comparison Table

Type	Interactive?	Trusted Setup?	Proof Size	Verification Speed	Scalability	Key Features / Use Cases
Interactive ZKP	Yes	No	Moderate	Moderate	Low (multi-round)	Foundational theory; secure voting, authentication
NIZK	No	Often Yes (CRS)	Small–Moderate	Good	Moderate	Offline verification; digital signatures, blockchain
zk-SNARK	No	Yes (per-circuit)	Very Small	Very Fast	High	Privacy coins (Zcash), DeFi, layer-2 scaling
zk-STARK	No	No	Larger	Fast	Very High	Transparent, quantum-resistant; StarkWare rollups
Bulletproofs	No	No	Small	Moderate–Slow (large)	Moderate	Confidential transactions (Monero), range proofs
Sigma Protocol	Yes	No	Small	Moderate	Low	Building blocks; authentication,

						Fiat–Shamir transform
PLONK	No	Yes (universal, updatable)	Very Small	Very Fast	High	Versatile SNARK; used in zkSync, Aztec, DeFi systems

Here are conceptual examples to help you intuitively understand the zero-knowledge proofs (ZKPs) at different levels without going into the complex theories and advanced mathematics behind them.

Examples

Example 1: The Magic Door Puzzle (Classic Example)

Alice knows the secret word that opens a magic door inside a cave. Bob wants to be sure she knows it, but Alice doesn't want to tell him the word.

- The cave has two tunnels, **A** and **B**, connected by the magic door.
- Alice walks into the cave while Bob waits outside. Bob doesn't know which tunnel she took.
- Bob then asks her to come out through **A** or **B**.
- If Alice knows the word, she can open the door and come out wherever Bob asks.
- If she doesn't, she can only guess—and will eventually get caught.

This proves she knows the secret **without revealing it**.

Example 2: A Color-blind friend and Two balls :

- There are two friends Bob and Alice, of whom Alice is color blind.
- Bob has two balls and he needs to prove that both balls are of different colours.
- Alice switches the balls randomly behind her back and shows it to Bob who has to tell if the balls are switched or not.
- If the balls are of the same color and Bob gives false information, the probability of him answering correctly is 50%.
- When the activity is repeated several times, the probability of Bob giving the correct answer with the false information is significantly low.
- Here Bob is the "prover" and Alice is the "verifier".

- Color is the absolute information or the algorithm to be executed, and its soundness is proved without revealing the information that is the color to the verifier.

Practical examples: Discrete log of a given value

These ideas can be applied to a more realistic cryptography application. Alice wants to prove to Bob that she knows the discrete logarithm of a given value in a given group. (Source)

For example, given a value *y*, a large prime *p*, and a generator gg, she wants to prove that she knows a value *x* such that $g^x \equiv y \pmod{p}$, without revealing *x*. Indeed, knowledge of *x* could be used as a proof of identity, in that Alice could have such knowledge because she chose a random value *x* that she did not reveal to anyone, computed $y = g^x \bmod p$, and distributed the value of *y* to all potential verifiers, such that at a later time, proving knowledge of *x* is equivalent to proving identity as Alice.

The protocol proceeds as follows: in each round, Alice generates a random number *r*, computes $C = g^r \bmod p$ and discloses this to Bob. After receiving *C*, Bob randomly issues one of the following two requests: he either requests that Alice disclose the value of *r*, or the value of $(x + r) \bmod (p - 1)$.

Bob can verify either answer; if he requested *r*, he can then compute $g^r \bmod p$ and verify that it matches *C*. If he requested $(x + r) \bmod (p - 1)$, then he can verify that *C* is consistent with this, by computing $g^{(x + r) \bmod (p - 1)} \bmod p$ and verifying that it matches $(C \cdot y) \bmod p$. If Alice indeed knows the value of *x*, then she can respond to either one of Bob's possible challenges.

If Alice knew or could guess which challenge Bob is going to issue, then she could easily cheat and convince Bob that she knows *x* when she does not: if she knows that Bob is going to request *r*, then she proceeds normally: she picks *r*, computes $C = g^r \bmod p$, and discloses *C* to Bob; she will be able to respond to Bob's challenge. On the other hand, if she knows that Bob will request $(x + r) \bmod (p - 1)$, then she picks a random value r', computes $C' \equiv g^{r'} \cdot (g^x)^{-1} \bmod p$, and discloses C' to Bob as the value of *C* that he is expecting. When Bob challenges her to reveal $(x + r) \bmod (p - 1)$, she reveals r', for which Bob will verify consistency, since he will in turn compute $g^{r'} \bmod p$, which matches $C' \cdot y$, since Alice multiplied by the modular multiplicative inverse of *y*.

However, if in either one of the above scenarios Bob issues a challenge other than the one she was expecting and for which she manufactured the result, then she will be unable to respond to the challenge under the assumption of infeasibility of solving the discrete log for this group. If she chooses *r* and disclosed $C = g^r \bmod p$, then she will be unable to produce a valid $(x + r) \bmod (p - 1)$ that would pass Bob's verification, given that she does not know *x*. And if she picked a value r' that poses as $(x + r) \bmod (p - 1)$, then she would have to respond with the discrete log of the value that she disclosed – but Alice does not know this discrete log, since the value C she disclosed was obtained through arithmetic with known values, and not by computing a power with a known exponent.

Thus, a cheating prover has a 0.5 probability of successfully cheating in one round. By executing a large-enough number of rounds, the probability of a cheating prover succeeding can be made arbitrarily low.

To show that the above interactive proof gives zero knowledge other than the fact that Alice knows x, one can use similar arguments as used in the above proof of completeness and soundness. Specifically, a simulator, say Dave, who does not know x, can simulate the exchange between Alice and Bob by the following procedure. Firstly, Dave randomly flips a fair coin. If the result is "heads", then he picks a random value r, computes $C = g^r \bmod p$, and discloses C as if it is a message from Alice to Bob. Then Dave also outputs a message "request the value of r" as if it is sent from Bob to Alice, and immediately outputs the value of r as if it is sent from Alice to Bob. A single round is complete. On the other hand, if the coin flipping result is "tails", then Dave picks a random number r', computes $C' = g^{r'} \cdot y^{-1} \bmod p$, and discloses C' as if it is a message from Alice to Bob. Then Dave outputs "request the value of $(x + r) \bmod (p - 1)$" as if it is a message from Bob to Alice. Finally, Dave outputs the value of r' as if it is the response from Alice back to Bob. A single round is complete. By the previous arguments when proving the completeness and soundness, the interactive communication simulated by Dave is indistinguishable from the true correspondence between Alice and Bob. The zero-knowledge property is thus guaranteed.

Homomorphic Encryption

Homomorphic encryption is a cryptographic method that allows computations, calculations and analytics to be performed on encrypted data without having to decrypt it first.

This means that data can then be shared with third parties in various industries who need to run the data through various algorithms, analyze or otherwise manipulate the data, without sharing the contents of the data.

Types of Homomorphic Encryption

Partially Homomorphic Encryption (PHE)

- Only **one** mathematical action or operation (addition or multiplication) can be performed on the encrypted values or data, however, this action can be performed unlimited times.
- A use case for this is the RSA public key encryption that is used for secure data transmission and this system relies on multiplication of two large prime numbers. The RSA system is an example of the multiplicative partially homomorphic encryption.

Somewhat Homomorphic Encryption (SHE)

- This type of HE supports **both** mathematical operations of addition or multiplication, however, only limited amounts of operations can be computed before noise, or complexity, becomes too much.
- An example of this is the Brakerski-Gentry-Vaikuntanathan (BGV) scheme.

Fully Homomorphic Encryption (FHE)

- The most powerful type of homomorphic encryption, this type supports **both** mathematical operations of addition or multiplication, **and** these actions can be performed unlimited times.
- An example of this is Gentry's work and improvements on homomorphic encryption. It allows for both additive and multiplicative operations on ciphertexts and this allows for the construction of circuits for performing different computations.

Applications

Applications of HE include:

- Analytics outsourcing: if a user wants to outsource computation but doesn't trust the third party with raw data, with HE, the data can be encrypted and sent to the third party who computes over encrypted data and returns an encrypted result; the user decrypts the result and the third party never sees the data

- Healthcare: HE allows aggregation or statistical operations on encrypted patient records
- Elections: Individual votes can be encrypted with HE and only the total count would need to be decrypted. This keeps the individual votes and choices secret/encrypted
- Database query: Allows a user to query a database without revealing the entirety of the database to the user
- Blockchain (and confidential transactions): In a blockchain setting, HE can be used to hide transaction amounts and other details while still allowing for validation and other computations.

Downsides of Homomorphic Encryption

- Computational overhead: As homomorphic encryption computation relies on computing very mathematical operations on large and encrypted numbers or polynomials, it is very intensive and causes a lot of delays in the processing of the information. This can make homomorphic encryption, especially FHE, very impractical.
- Noise growth: Noise in homomorphic encryption can be pictured as a small integer during the computations. Because there are very large computations being completed, the more computations there are, the more the noise grows, making it hard to ignore. If the noise grows to a certain level, it will be impossible to decrypt the ciphertext to get back the original value.

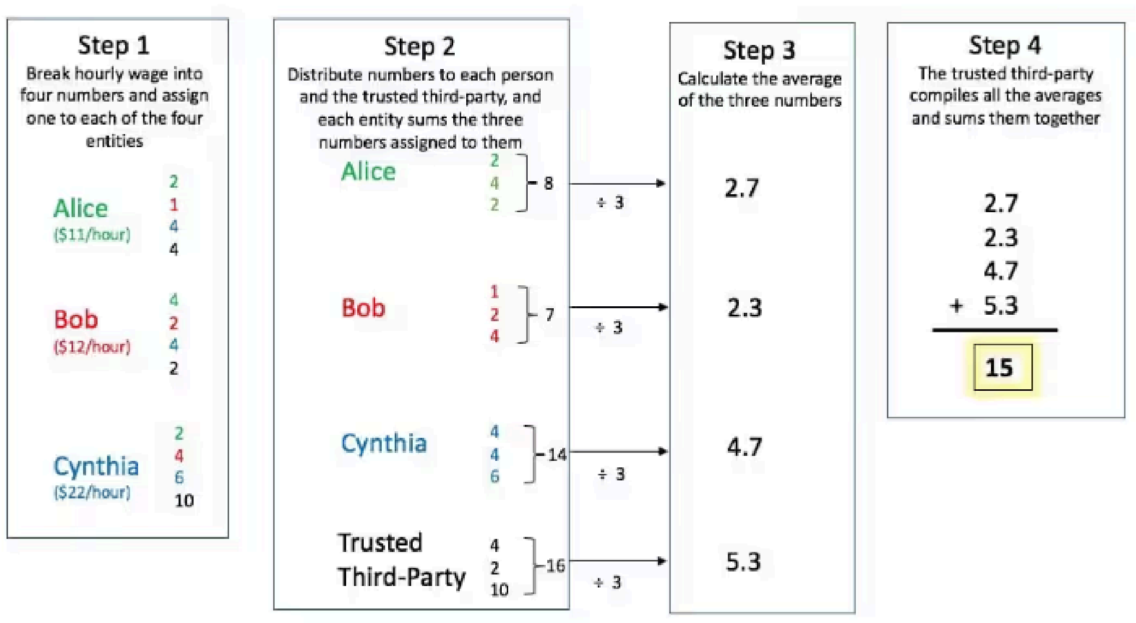
Multi-Party Computation

Introduction
First, we must let you know that this teaching aide doesn't fully cover the complexity that is Multi-Party Computation and as such, further reading may be required, should your interest in this technique be sparked. There's a lot more to Multi-Party Computation than the basic idea, and "Gartner expects SMPC to be transformational in the next 5-10 years (Lowans, 2020)". Feel free to check out the bibliography section below for further reading as needed.

The concept of Multi-Party Computation (MPC) was first introduced by Andrew Yao in the 1980s, and its direct meaning is multiple parties coming together to compute a function, where only the output is known by all of them, and their inputs are a secure form of their raw data. The whole concept of MPC is of course mathematical, since it literally is a function that multiple parties come together to compute. While the intricacies of the function will not be explored in this teaching aide, feel free to have an insightful read via ScienceDirect. That said, we will still explore an example that helps apply MPC to the real world (via Chainlink):

In a typical SMPC protocol, each party holds a piece of private data and wants to compute a function that requires inputs from all parties. Through the protocol, parties use privacy-preserving techniques to exchange input data, such as encryption or masked shares, and then collectively compute the function.

Calculating an Average of Hourly Wages



Step 1
Break hourly wage into four numbers and assign one to each of the four entities
Alice ($11/hour) 2 1 4 4
Bob ($12/hour) 4 2 4 2
Cynthia ($22/hour) 2 4 6 10
Step 2
Distribute numbers to each person and the trusted third-party, and each entity sums the three numbers assigned to them
Alice 2 4 2 8
Bob 1 2 4 7
Cynthia 4 4 6 14
Trusted Third-Party 4 2 10 16
÷ 3
÷ 3
÷ 3
÷ 3
Step 3
Calculate the average of the three numbers
2.7
2.3
4.7
5.3
Step 4
The trusted third-party compiles all the averages and sums them together
2.7
2.3
4.7
+ 5.3
15

Imagine that three coworkers Alice, Bob, and Cynthia, want to know their average hourly wage but don’t want to share their own salary with each other. First, they break their wage into four amounts that add up to their hourly earnings. Next, they keep one of those figures, and share one each with the other coworkers along with a trusted third party. Now, each party calculates the average of the numbers they received. Finally, these averages are then shared and summed to provide the average hourly wage. While they all know the average, they don’t know each other’s individual earnings.

While this example uses a relatively simple additive secret-sharing technique, you can explore a more advanced mathematical example here.

Sometimes called Secure Multi-Party Computation (SMPC), MPC is a cryptographic and mathematical concept that ensures that everyone in a function is able to keep their raw data unknown to other parties of the group. The computation that makes up this technique mandates that parties first protect their raw data through a method like encryption before then submitting them as input for the MPC function to be combined with that of other parties. While the output is known and glaring for the group, that value cannot be used to trace back the original unencrypted input.

SMPC enables “black box” functionality where many people can work on a calculation together using their private information. Even though everyone can see the result, their data is kept secret. (Source)

Advanced Definition (Source)

SMPC uses cryptographic primitives like secret sharing (e.g. Shamir), homomorphic encryption (e.g. Paillier, ElGamal), and zero-knowledge proofs (e.g., zk-SNARKs, zk-STARKs) to enable a given number (n) of participants each with private data (d1, d2, …, dn) to compute a public function on that data F(d1, d2, …, dn), without any participants learning information about another's input.

SMPC protocols must ensure:

- **Privacy**: No party can see or deduce private inputs from any other party.
- **Accuracy**: Parties that deviate from the protocol instructions cannot force honest participants to output an incorrect result.

The Importance of SMPC

MPC is an important component for executing or analyzing data-sensitive processes such as financial transactions, medical research, distributed voting, private bidding and auctions, AI/ML learning processes, etc. It is ultimately useful in areas where sensitive information needs to be processed or analyzed. It allows users the ability to safeguard their private data while still actively partaking in shared systems.

Among other proprietary products and information that are termed a company's crown jewel, organizations are understandably and increasingly concerned about data security in a myriad of scenarios, including:

- The collection and maintenance of personal information, which is inherently sensitive by nature,
- The process of storing, using and working with personal information in external environments, e.g. the cloud,
- Sharing and handling sensitive data in different industries, from healthcare to finance.

MPC ensures collaboration between systems in the society without compromising sensitive data. It alleviates the concern rooted in the confidentiality and integrity of information especially when they've been shared with multiple parties.

Use Cases

MPC/SMPC is very popular for scenarios where organizations need to collaborate with other players without the need to disclose or reveal their proprietary information to the players.The following list looks at the different use cases that organizations tend to employ MPC for.

- Collaborating with parties regardless of their differing structure and/or policies, e.g., sharing citizen data between government departments and/or financial institutions; or exchanging electronic medical records amongst hospitals, pharmacies, insurance manufacturers
- Collective data sharing, where the private data knowledge is needed from independent data sources to learn something that they would otherwise be unable to obtain or access from a single source, for example, verifying the travel history of a visa applicant by collaborating with other countries, or obtaining criminal history information as stated in the visa application
- Key management use cases which involve safeguarding authentication keys as they are being used, making them dynamic enough to avoid adversaries making any plans with their knowledge
- Data exchange through cloud computing, obtaining data analytics, and Machine Learning across multiple, unknown cloud providers
- Private data aggregation via monitoring entities' multiple network security practices
- Secure email practices like spam filtering to prevent unauthorized entry via business email compromise, for example
- Medical discovery, e.g., disease or virus contact tracing apps, combining data of many hospitals for genomics research

Benefits of Multi-Party Computation

MPC provides various benefits for anyone using it to ensure privacy-reserving cryptography, and this could be anyone including private individuals, developers, organizations, research groups, etc:

- **Enhanced Security:** MPC offers a much effective protection against data breaches by maintaining data confidentiality throughout the computation process. For MPC, the priority of secure processes focuses on the input rather than the output since the raw data need not be known, and as such, confidentiality of a party's data is kept consistent throughout.
- **Data privacy:** Parties can retain the confidentiality of their data while confidently collaborating with other parties for use of their data applications, research, voting mechanisms, etc.

- **Regulatory compliance:** Organizations can continuously adhere to data protection regulations (e.g., GDPR, HIPAA) while storing, handling and processing sensitive data without fear of exposure, which ultimately curtails the risk of non-compliance.
- **Collaboration:** With MPC, multiple parties (involving organizations and competitors alike, or even institutions in different industries) can be assured of being able to securely aggregate and analyze data from different sources without damaging privacy, which in turn champions cross-organizational partnership and knowledge sharing.
- **Accuracy:** As MPC produces effective results, any outcomes from collaborations promises accuracy levels required for high-value use cases.
- **Quantum-safe:** Encrypting pieces of data before then computing a function with them by collaborating with multiple parties via a cryptographic function is akin to breaking up data and distributing it among participants. To avoid data breaches, a computational function must be immune to quantum attacks, and the very set up of MPC allows for this.
- **Advanced functionality:** Applications can provide users with advanced functionality without compromising on security.

MPC Techniques

As a privacy preserving technique itself, MPC does have two main leading techniques that are highly adopted today: *Garbled Circuits* and *Secret Sharing.*

Garbled Circuits: here, “one party encrypts each input bit to create a “wire label”. Then, it converts the computation into a circuit of binary gates, each of which are expressed as a “garbled truth table” made up of a few ciphertexts”. One party; referred to as the evaluator, receives the binary gates circuit as well as the created wire labels. They then combine the inputs to produce an encrypted outcome.

While evaluation is ongoing, the garbled truth table circuits do not require any form of communication between parties, however their state is far larger than the input data (e.g. 80 -128^x for typical security parameters).

Secret Sharing: in contrast, this splits each sensitive *integer* input into something called “secret shares”. Once these are combined, they produce the original data. In a typical encoding scheme, the original data is revealed once you add the secret shares together.

Differential Privacy (DP)

Introduction

Differential Privacy (DP) represents a mathematical framework that Dwork created in 2006 to protect personal data while enabling useful statistical analysis (Dwork, 2008). The fundamental principle of DP ensures that adding or removing individual data points from a dataset produces only small variations in analysis results. The method of DP provides quantifiable security through noise addition to calculations which protects against re-identification threats that traditional anonymization techniques face (Dwork, 2008; NIST, 2025). The National Institute of Standards and Technology (NIST) identifies DP as the most effective protection method because it maintains strong security even when adversaries possess additional information (NIST, 2025).

Principles of Differential Privacy (DP)

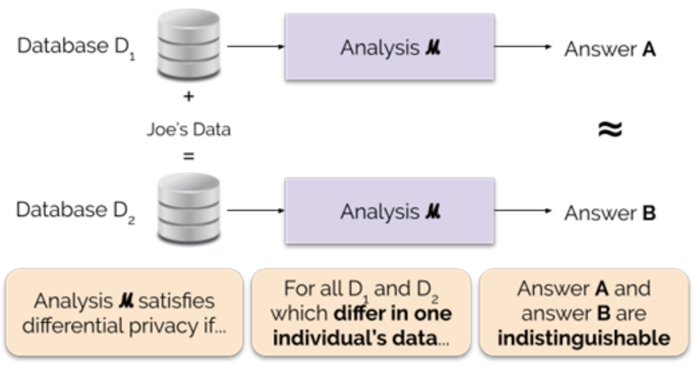
I. Mathematical Definition

A randomized algorithm M is said to satisfy (ε, δ)-differential privacy if, for any two datasets that differ by one individual, the likelihood of M yielding a specific output varies only minimally:

$$\Pr[M(D1) \in S] \leq e^{\varepsilon} \cdot \Pr[M(D2) \in S] + \delta$$

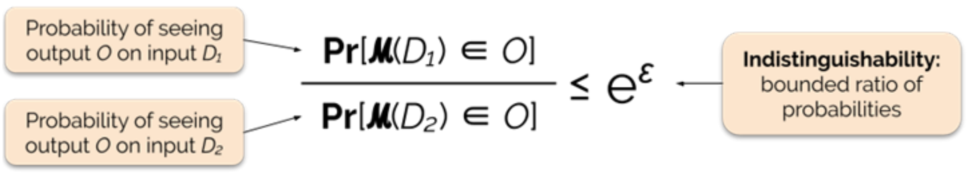
In this context, ε (epsilon) governs the privacy loss (lower values signify enhanced privacy), whereas δ permits a slight chance of failure (Dwork, 2008).

In informal terms, differential privacy ensures that for every person who shares data for analysis, the result of a differentially private analysis remains approximately unchanged, regardless of whether you provide your data. An analysis that ensures differential privacy is commonly referred to as a mechanism, which we represent as $\mathcal{M}$


Database D_1
Analysis $\mathcal{M}$
Answer **A**
+
Joe's Data
=
≈
Database D_2
Analysis $\mathcal{M}$
Answer **B**
Analysis $\mathcal{M}$ satisfies differential privacy if...
For all D_1 and D_2 which **differ in one individual's data**...
Answer **A** and answer **B** are **indistinguishable**

Informal Definition of Differential Privacy (Near, J et, al., 2020)

The concept is illustrated through Figure 1. The calculation of Response "A" happens without Joe's data yet Response "B" includes his information. The privacy model of differential privacy requires the two responses to have identical characteristics. The output reveals no information about whether Joe's data entered the system or what specific data points he contributed. The privacy assurance strength depends on the privacy parameter ε which serves as the privacy loss or privacy budget. The protection of individual data becomes stronger when the ε parameter value decreases because it produces results that are less distinguishable (Near, J et, al., 2020)


Probability of seeing output O on input D_1
Probability of seeing output O on input D_2
$\frac{\Pr[\mathcal{M}(D_1) \in O]}{\Pr[\mathcal{M}(D_2) \in O]} \leq e^{\varepsilon}$
Indistinguishability: bounded ratio of probabilities

Formal Definition of Differential Privacy (Near, J et, al., 2020)

The addition of random noise to answers serves as a method to achieve differential privacy when responding to queries. The main challenge lies in determining both the points for noise insertion and the appropriate amount of noise to add. The Laplace mechanism stands as a common technique for adding noise to data. The amount of noise added to sensitive queries increases with their sensitivity level to achieve the desired `epsilon` level of differential privacy which might decrease the usefulness of the results.

II. Adding Noise

The Laplace mechanism and Gaussian mechanism serve as standard methods to add noise to query results which protects individual privacy by making personal information unrecoverable (Dwork, 2008; NIST, 2025).

III. Adversarial Model

Differential Privacy (DP) assumes adversaries may have access to unlimited background knowledge. Unlike k-anonymity or l-diversity, DP is resistant to linkage and differencing attacks (Dwork, 2008).

Applications

1. **National Statistics:** The U.S. Census Bureau employed DP during its 2020 census to protect participant privacy while delivering statistical data (Alborch Escobar et al.,2024)
2. **Machine Learning:** DP implementation in federated learning defends against membership inference attacks by using Differentially Private Stochastic Gradient Descent (DP-SGD) which introduces random noise to model gradient updates (NIST, 2025).
3. **Blockchain and Finance:** The implementation of verifiable local differential privacy in blockchain financial operations allows statistical analysis through unlinkable data processing (Movsowitz Davidow et al. 2023)
4. **Privacy-Enhancing Technologies:** Privacy-Enhancing Technologies depend on Differential Privacy as their fundamental component because this technology operates with Fully Homomorphic Encryption and Secure Multi-Party Computation and Trusted Execution Environments according to (Belorgey & Carpov. 2024).
5. **Encrypted Databases:** The integration of DP with encryption through CDP enables users to run privacy-preserving database queries on encrypted cloud storage systems according to (Alborch Escobar et al.2024)

Advantages

- The system achieves better protection against attacks when auxiliary information is integrated into the system (Dwork, 2008).
- The system provides mathematically proven privacy protection which users can independently verify (Dwork, 2008).
- The system demonstrates its ability to handle big data volumes and enables artificial intelligence operations and federated learning and distributed architecture management according to NIST (2025).

Disadvantages

- Utility versus Privacy Trade-off: The system's ability to handle utility and privacy trade-offs becomes more difficult because data precision decreases when the system receives excessive noise. (NIST 2025)
- Parameter Adjustment: DP systems face dual technological and social obstacles when users need to select ε parameters for their systems. (Dwork 2008)
- Computational Expenses: DP with homomorphic encryption for cryptographic methods needs strong computational resources to operate according to Alborch Escobar et al. (2024).
- Misuse: The implementation of DP principles fails to produce results when organizations use them improperly according to NIST 2025.

Emerging Techniques

- Encrypted database Computational Differential Privacy (CDP) (Alborch Escobar et al., 2024).
- The use of Verifiable Local Differential Privacy (Movsowitz Davidow et al., 2023) for securing blockchain transactions.
- The combination of differential privacy with secure multiparty computation and homomorphic encryption represents a new approach to privacy protection (Belorgey & Carpov, 2024).
- The NIST SP 800-226 (NIST, 2025) provides official assessment criteria for DP systems.

The field of privacy-preserving cryptography now relies on Differential Privacy as its core technology which protects sensitive information in national statistics and machine learning applications and financial systems and cloud computing environments. The integration of differential privacy with cryptographic methods shows promise for future data security because it addresses ongoing difficulties in maintaining privacy levels while delivering useful data (Dwork, 2008; NIST, 2025).

Anonymous Credentials and Commitment Schemes

Modern digital services rely heavily on identity information, but the way most systems are built forces people to share more personal data than is necessary. This creates obvious risks. Once collected, data can be leaked or misused. Hence why anonymous credential systems were introduced as a solution. It gives users the ability to prove only what is needed, instead of who they are. For example, users can prove “I am over 18” or “I have a valid subscription”, without revealing their identity.

The challenging aspect is how to make this secure. The system must be able to support unlinkability and selective disclosure, along with the ability to resist forgery and credential sharing between users. Since the 1980s, scientists have put together the building blocks to make it possible. The 2 building blocks are, commitment protocols (users commit values without the need of revealing the value to anyone) and zero-knowledge proofs (helps prove something without revealing information about it). These ideas were then integrated into practical protocols, such as the well-known Camenisch Lysyanskaya(CL01) scheme, and subsequently into real systems such as IBM's Idemix and Microsoft's U-Prove.

The Origin: Chaum (1985)

David Chaum developed the idea of doing transactions and identification activities without identification in 1985, thereby enabling people to use services and prove rights without exposing their real identity. The motivation behind it was user privacy, it lets users verify without revealing extra information about themselves. Revealing extra information is unsafe as it could be used to identify them or link them to other actions. David Chaum brought this concept both as a conceptual framework and as a set of baselines which would enable such interactions.

An Example

Suppose we place a paper in a sealed envelope containing a carbon copy. Once we seal the envelope, we make trusted issuer sign the envelope without him/her opening it, hence a blind signature. After some time, we open the envelope and the issuer's signature still holds good for the paper. Since the issuer never knows what was in it, the issuer can't associate the signature with the party who will use the signed paper later. David Chaum used this to show how credentials can be published and then presented unlink ably so the user can control when and where the credential is revealed.

Fundamental building blocks

Commitment Schemes

A commitment scheme allows the users to commit to a value at the time and reveal it at a later time. The user can not change the value he committed to (binding) and the receiver will not know anything about the value until the user reveals it. As seen, commitment schemes have 2 main parts, committing & binding.

There are two phases in a commitment scheme -

1. Commit Phase: The user computes C such that C = Commit (m,r), where m is the secret value and r is a random value, the user then sends C to the issuer/verifier.
2. Reveal Phase: The user later reveals the secret value m and the random value r. The verifier confirms if C = Commit (m,r).

Integration with Anonymous Credentials:

- The issuer signs unseen committed attributes..
- The user can provide proof over the commitment using zero-knowledge proofs, thus preserving both unlinkability and privacy.

Zero-Knowledge Proofs (ZKPs)

Zero Knowledge Proofs allows users to prove to a verifier that a statement is true without revealing any information about it (Camenisch & Lysyanskaya 2001 (CL01)). In anonymous credentials, ZKPs is used by users to prove possession of a credential or show that an attribute satisfies a condition while keeping the attribute hidden. Commitment schemes & Zero Knowledge Proofs compliment each other well by enabling multishow capabilities and selective disclosure while maintaining unlink ability.

In summary, the steps involved are -

1. Commit attribute → C = Commit (m, r)
2. Issuer signs C → hence credential is issued
3. User generates ZKP proving predicate on attribute m
4. Verifier checks ZKP → accepts it if valid

The Next Big Step : LRSW (1999)

In 1985, David Chaum laid the groundwork for anonymous credentials, but the next milestone came in 1999 when Lysyanskaya, Rivest, Sahai, and Wolf (LRSW) introduced the first formal security model. Instead of just describing a system, the rules and guarantees that such credentials need to have to be actually secure were defined by them. They had attempted to formalize how users obtain, and use unlinkable attributes across services, and they proposed cryptographic assumptions and constructions for achieving unlinkability. The LRSW model debunked the important properties any anonymous credential system needs to satisfy. Such as - it must be non-transferable (only the valid user can use it), unforgeable (nobody can create false credentials), unlinkable (separate uses of the same credential cannot be tied together), and provide selective disclosure (users reveal only what is necessary). While the 1999 paper did not yet have a fully functional system to show, it provided the conceptual and mathematical kit that researchers needed to design future schemes. This directly inspired the development of Camenisch–Lysyanskaya (CL01) in 2001. In brief - LRSW gave the industry a formal model to reason about privacy.

Camenisch–Lysyanskaya (CL01, 2001)

Then in 2001, Camenisch & Lysyanskaya introduced a breakthrough. They introduced an efficient and non-transferable anonymous credential system. In their system, users can repeatedly present the same credential to as many verifiers as they wish, without any interaction with the issuer and without the presentations being linkable. They also discussed selective disclosure of, as in the holder only reveals the requested attributes or proves predicates of them. Their protocol also allowed anonymity to be revoked optionally, so in case of abuse, the protocol can reveal the identity of the user under controlled conditions (this was a trade-off between privacy and accountability). So by combining commitment schemes, special signatures, and Zero Knowledge Proofs, CL01 had rendered the system efficient enough to be implemented. They turned academic ideas into deployable protocols.

How CL01 works

Assume that an issuer provides a user with a signed credential over his committed attributes. To demonstrate something, the user calculates a Zero Knowledge Proof that he has a valid signature over committed attributes satisfying the verifier's predicate. The ZKP hides the attribute values and uses cryptographic bindings for non-sharing and non-forging. Revocation, if needed, is achieved by techniques allowing authorities to open/revoke a presentation under strict conditions.

Real Words Applications

Idemix (by IBM)

IBM’s Idemix adds API layers and deployment options for identity systems. Idemix also offers selective disclosure, unlinkable presentations, and practical ways to revoke credentials. It shows how the CL theoretical design translates into software components that developers can use.

U-Prove (by Microsoft)
Microsoft’s U-Prove has a different design focus. It highlights efficiency and straightforward proofs while still ensuring unlinkability for presentations. U-Prove tokens encode attributes and are created to be issued and presented with minimal computational overhead. This presents a useful contrast to CL Idemix in a class discussion about trade-offs, such as feature richness versus runtime cost.

Modern improvements

IBM’s Idemix and Microsoft’s U-Prove were early attempts to turn anonymous credentials into real-world tools. Idemix was considered very powerful but too complicated, it was great in research, but too heavy for everyday use. While U-Prove was faster and simpler, once Microsoft dropped its identity platform, it never caught on.

Still, both left a mark. They showed that privacy-friendly credentials weren’t just theory, and many of their ideas are now baked into newer systems like decentralized identity wallets and W3C Verifiable Credentials. Recent work focuses on making credential systems faster, more compact and better at revocation.

Anonymous credentials remain relevant to new fields of applications like decentralized identity systems, privacy-preserving e voting, and digital purses. New techniques such as pairing based cryptography, accumulators, and reduced-zero knowledge proofs will become mainstream and anonymous credentials will persist to develop, allowing users to prove their rights online without revealing their identity.

Challenges to the Application of Privacy Preserving Cryptography

Sometimes, even the best mathematical computations can be found to have some flaws, either in their design or application. There may also sometimes be other limiting factors that contribute a challenge to those adopting them. Listed below are a few challenges facing the adoption of Privacy Preserving Cryptography as a whole:

Performance and Scalability: advanced cryptographic techniques like Homomorphic Encryption pose a computational overhead challenge, as they allow intensive computations that require complex data processing requirements. The overhead makes it quite challenging to scale effectively especially for situations where the use of large datasets are needed.

Implementation Issues: in most, if not all computational algorithms, bugs and errors will exist. Unfortunately, PPC is not immune from being vulnerable to these limitations. Though algorithms thrive better with constant updates and fixes, a bug-infested implementation does expose the software to vulnerabilities and there is no definite guarantee that these errors will be completely resolved.

Key Management Risk: In most cases, keys may just be as sensitive as the information they're used to secure and protect. That said, poor and insecure key management practices make the entire system vulnerable to security breaches and can weaken the entire system.

Multi-Company Data Exchange: Because PPC does allow for company collaboration regardless of their diverse industry backgrounds, these companies will be open to trading varying data that require unique computational capabilities. The complexity required to process these kinds of data can complicate the design and deployment of privacy-preserving solutions.

Decryption in Servers: In many applications, data must be decrypted for server-side processing, creating a window of vulnerability where the data is exposed to threats on the third-party server, according to ScienceDirect.

Quantum Computing: Though a quality of PPC is that it's quantum safe, the advancement of powerful quantum computers could still threaten to break current public-key cryptography algorithms like RSA and Diffie-Hellman, creating a potential security risk that constrain the development of quantum-resistant cryptography.

System Integration Complexity: Assimilating advanced privacy-preserving cryptographic tools like the Internet of Things (IoT), web browsing, etc into large-scale, real-world operations poses a complex technical problem.

Conclusion

Privacy transcends companies, it’s woven into data and information, and from storage, to access, to usage, and to the general application of data/information, people are at its heart. Privacy continues to be a tricky subject to truly manage, and Privacy Preserving Cryptography is only one of the steps to having privacy truly **be** private. With its many faces (techniques), as well as its privacy preserving makeup, PPC is another proof that cryptographic applications and its evolutions are great choices for cybersecurity and privacy as a whole. There’s a lot of hope yet for individuals and companies moving closer to secure systems and practices that foster efficient privacy and security.

Bibliography

Sheybani, Nojan, et al. “Zero-knowledge Proof Framework: A Systematic Survey”, 27 Apr. 2015. arxiv.org/pdf/2502.07063

Jayodya Methmal, “Zero Knowledge Proofs: A Comprehensive Review of Applications, Protocols, and Future Directions in Cybersecurity”, August 2023. DOI:10.13140/RG.2.2.11606.22080

A. Pathak, T. Patil, et al. "Secure Authentication using Zero Knowledge Proof," 2021 Asian Conference on Innovation in Technology (ASIANCON), PUNE, India, 2021, pp. 1-8. DOI: 10.1109/ASIANCON51346.2021.9544807

Lu Zhou, Abebe Diro, et al. “Leveraging zero knowledge proofs for blockchain-based identity sharing: A survey of advancements, challenges and opportunities”, Journal of Information Security and Applications, Volume 80, 2024, 103678, ISSN 2214-2126. DOI: 10.1016/j.jisa.2023.103678

Zanussi, Zachary. “Privacy Preserving Technologies Part Two: Introduction to Homomorphic Encryption”. *Statistics Canada*, 01 March 2022. www.statcan.gc.ca/en/data-science/network/homomorphic-encryption
“What is homomorphic encryption?” *IBM*. www.ibm.com/think/topics/homomorphic-encryption. Accessed 16 September 2025.

“Homomorphic Encryption”. *Chainlink*, 6 June 2025. www.chain.link/education-hub/homomorphic-encryption

Paine, Kirsty. “Homomorphic Encryption: How It Works”. *Splunk*, 5 February 2024. https://www.splunk.com/en_us/blog/learn/homomorphic-encryption.html

Bryanton, Betty Ann. “Introduction to Privacy Enhancing Cryptographic Techniques: Secure Multiparty Computation”. *Statistics Canada*, 15 March 2024. www.statcan.gc.ca/en/data-science/network/multiparty-computation

“Secure Multi-Party Computation”. *Chainlink*, 14 August 2024. www.chain.link/education-hub/secure-multiparty-computation-mcp

Lindel, Yehudal. “Secure Multiparty Computation (MPC)”. *Unbound Tech and Bar-Ilan University*. eprint.iacr.org/2020/300.pdf. Accessed 16 September 2025.

Evans, David, et al. "A Pragmatic Introduction to Secure Multi-Party Computation". *now Publishers Inc*, 19 December 2018. www.nowpublishers.com/article/Details/SEC-019

Maurer, Ueli. “Secure Multi-Party Computation Made Simple”. *ScienceDirect*, 3 October 2005. www.sciencedirect.com/science/article/pii/S0166218X05002428

Volgushev, Nikolaj, et al. “Conclave: Secure Multi-Party Computation on Big Data”. *ACM Digital Library*, 25 March 2019. www.dl.acm.org/doi/abs/10.1145/3302424.3303982

Alborch Escobar, Ferran, et al. “Computational Differential Privacy for Encrypted Databases Supporting Linear Queries.” Proceedings on Privacy Enhancing Technologies, 2024. petsymposium.org/popets/2024/popets-2024-0131.pdf

Dwork, Cynthia. “Differential Privacy: A Survey of Results.” 2008. www.cs.ucdavis.edu/~franklin/ecs289/2010/dwork_2008.pdf

Mironov, Ilya. “On Significance of the Least Significant Bits for Differential Privacy.” University of Waterloo, crysp.uwaterloo.ca/courses/pet/F18/cache/Mironov.pdf

Chaum, David. “Security without Identification: Transaction Systems to Make Big Brother Obsolete“, October 1985. https://www.cs.ru.nl/~jhh/pub/secsem/chaum1985bigbrother.pdf

Damgård, Ivan & Nielsen, Jesper. “Commitment Schemes and Zero-Knowledge Protocols“, 2011. https://cs.au.dk/%7Eivan/ComZK06.pdf

Lysyanskaya, Anna & Rivest, Ronald & Sahai, Amit & Wolf, Stefan. “Pseudonym Systems“, 1999. https://crypto.ethz.ch/publications/files/LRSW99.pdf

Camenisch, Jan & Lysyanskaya, Anna. “An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation“, 2001. https://cs.brown.edu/people/alysyans/papers/cl01a.pdf

Camenisch, Jan & Herreweghen, Els. “Design and Implementation of the idemix Anonymous Credential System“. IBM Research. https://www.freehaven.net/anonbib/cache/idemix.pdf

Paquin, Christian. “U-Prove Technology Overview V1.1“, Microsoft, April 2013 https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/U-Prove20Technology20Overview20V1.120Revision202.pdf?msockid=19ca7afbd4c1646b36bd6c9cd54b6500

Camenisch, Jan & Lysyanskaya, Anna. “Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials“, February 2002. https://cs.brown.edu/people/alysyans/papers/camlys02.pdf

Alborch Escobar, F., Canard, S., Laguillaumie, F., & Phan, D. H. (2024). Computational Differential Privacy for Encrypted Databases Supporting Linear Queries. Proceedings on Privacy Enhancing Technologies, 2024(4), 583–604.

Belorgey, M. G., & Carpov, S. (2024). Combining Cryptography and Other Techniques for Various Privacy-Preserving Applications. NIST Crypto Reading Club, May 15, 2024.

Dwork, C. (2008). Differential Privacy: A Survey of Results. Lecture Notes in Computer Science, 4978, 1–19.

Dwork, C., & Roth, A. (2014). The Algorithmic Foundations of Differential Privacy. Foundations and Trends in Theoretical Computer Science, 9(3–4), 211–407.

Movsowitz Davidow, D., Manevich, Y., & Toch, E. (2023). Privacy-Preserving Transactions with Verifiable Local Differential Privacy. Tel Aviv University & IBM Research.

Near, J., Darais, D., & Boeckl, K. (2020, July 27). Differential privacy for privacy-preserving data analysis: An introduction to our blog series. Cybersecurity Insights (NIST blog). https://www.nist.gov/blogs/cybersecurity-insights/differential-privacy-privacy-preserving-data-analysis-introduction-our

NIST. (2025). Guidelines for Evaluating Differential Privacy Guarantees. NIST Special Publication 800-226. U.S. Department of Commerce.

Morris, Dana. “How Encryption Works to Preserve Data Privacy”. *Dataversity*, January 3, 2023. https://www.dataversity.net/articles/how-encryption-works-to-preserve-data-privacy/#:~:text=to%20have%20both.-,Privacy%2DPreserving%20Cryptography,-Privacy%2Dpreserving%20cryptography

Craddock, Mark et. al. “UN Handbook on Privacy-Preserving Computation Techniques”. *Big Data UN Global Working Group*, Accessed September 30, 2025. https://unstats.un.org/bigdata/task-teams/privacy/UN%20Handbook%20for%20Privacy-Preserving%20Techniques.pdf

“Privacy-Preserving Collaboration Using Cryptography”. *Digital.gov*, Accessed September 30, 2025. https://digital.gov/resources/privacy-preserving-collaboration-using-cryptography#:~:text=Secure%20multi%2Dparty%20computation%20(MPC)%20is%20a%20type%20of,optimize%20MPC%20for%20complex%20functions.