YOUR SCIENCE

CREATING OPPORTUNITY

Applied Analytics
AI & ML for BI & Automation

5. Data Protection and Privacy
in AI-Driven Manufacturing

Your Science
Mathematical Consulting
Prof. Norbert Poncin
2025

Your Science Center: 31, Boulevard Prince Henri, L-1724 Luxembourg
Phone: +352 621 674 917, **Email**: info@yourscience.eu, **Website**: yourscience.eu
Business Permit: 10154927, **LBR**: A44409, **VAT**: LU35024328

YOUR SCIENCE
CREATING OPPORTUNITY

Applied Analytics
AI & ML for BI & Automation

4. Data Protection and Privacy in AI-Driven Manufacturing

Your Science
Mathematical Consulting
Prof. Norbert Poncin
2025

Your Science Center: 31, Boulevard Prince Henri, L-1724 Luxembourg
Phone: +352 621 674 917, **Email**: info@yourscience.eu, **Website**: yourscience.eu
Business Permit: 10154927, **LBR**: A44409, **VAT**: LU35024328

Contents

1 Data Protection

1.1 General Data Protection Regulation

The **General Data Protection Regulation** (GDPR) is the European Union's primary law on data protection and privacy for individuals. Key points include:

- **Personal Data Definition**: The GDPR broadly defines personal data to include any information that can directly or indirectly identify an individual (e.g., name, email, IP address).

- **Lawful Processing**: Data can only be processed under specific grounds, such as:

  1. **Consent** (e.g., agreeing to a newsletter via a checkbox).
  2. **Contractual necessity** (e.g., processing employee bank details for salary payment).
  3. **Legal obligation** (e.g., storing tax-related data for compliance).
  4. **Vital interests** (e.g., sharing medical data during emergencies).
  5. **Public interest** (e.g., processing census data).
  6. **Legitimate interests** (e.g., fraud detection in e-commerce).

- **Consent**: Consent must be freely given, specific, informed, and unambiguous, with the option to withdraw at any time.

- **Data Subject Rights**: Individuals have the right to:

  1. **Access**: Know what personal data is held and how it's used.
  2. **Rectification**: Correct inaccurate data.
  3. **Erasure**: Request data deletion under certain conditions.
  4. **Portability**: Transfer data to another provider (e.g., moving photos between cloud services).
  5. **Objection**: Object to processing based on legitimate interests (e.g., against data being used for product improvement or marketing).

- **Data Minimization & Purpose Limitation**: Collect only the data necessary for a specific purpose and avoid further use without consent.
- **Accountability & Transparency**: Organizations must demonstrate compliance and provide clear, accessible privacy notices.
- **Data Protection by Design and Default**: Integrate data protection into business processes from the outset.
- **Data Breach Notification**: Notify the supervisory authority within 72 hours of a breach affecting individuals' rights.
- **Data Protection Officer** (DPO): Large organizations or those processing sensitive data must appoint a DPO to oversee compliance.
- **International Transfers**: Transferring personal data outside the EU requires adequate protection measures in the receiving country.
- **Penalties**: Non-compliance can result in fines up to 20 million euros or 4% of global annual revenue, whichever is higher.

1.2 California Customer Privacy Act

The **California Customer Privacy Act** (CCPA) is another prominent regulation. In Table 1 the reader finds a comparison between the GDPR and the CCPA.

1.3 Recommended Measures for Data Privacy Compliance

Ensuring adherence to GDPR necessitates implementing measures across multiple organizational domains. The following recommendations are categorized by their respective implementation levels.

- **CRM Systems:**
  - **Clear Privacy Notices**: Provide concise, transparent privacy notices outlining what data is collected, how it is processed, and with whom it is shared.

Table 1: Key differences between GDPR and CCPA

Aspect	**GDPR**	**CCPA**
Scope	Applies to all entities worldwide that process the personal data of EU residents.	Applies to for-profit businesses meeting size or data thresholds and handling California residents' data.
Personal Data Definition	Broad, includes any information identifying a person (e.g., IP addresses, biometrics).	Includes similar categories but explicitly adds household data and consumer behavior.
Consent	Requires explicit consent for data collection and processing.	Focuses on notice and opt-out rights, particularly for data sales.
Rights Granted	Includes data portability, correction, and erasure.	Focuses on access, deletion, and opt-out of data sales, with no correction rights.
Penalties	Fines up to € 20M or 4% of annual global turnover, whichever is higher.	Fines up to $ 7,500 per intentional violation.
Focus	Comprehensive personal data protection and privacy.	Consumer rights and transparency, especially around data sales.

  - **Informed Consent**: Ensure explicit, informed consent is obtained before processing data, with a straightforward mechanism for users to withdraw consent.
  - **Marketing Opt-Out**: Include an easily accessible 'unsubscribe' option in all marketing communications.
  - **Data Access and Portability**: Facilitate user access to their personal data and provide the ability to download and transfer (port) data to another service provider.
  - **Consequences of Data Deletion**: Clearly inform users of the implications of deleting their data.

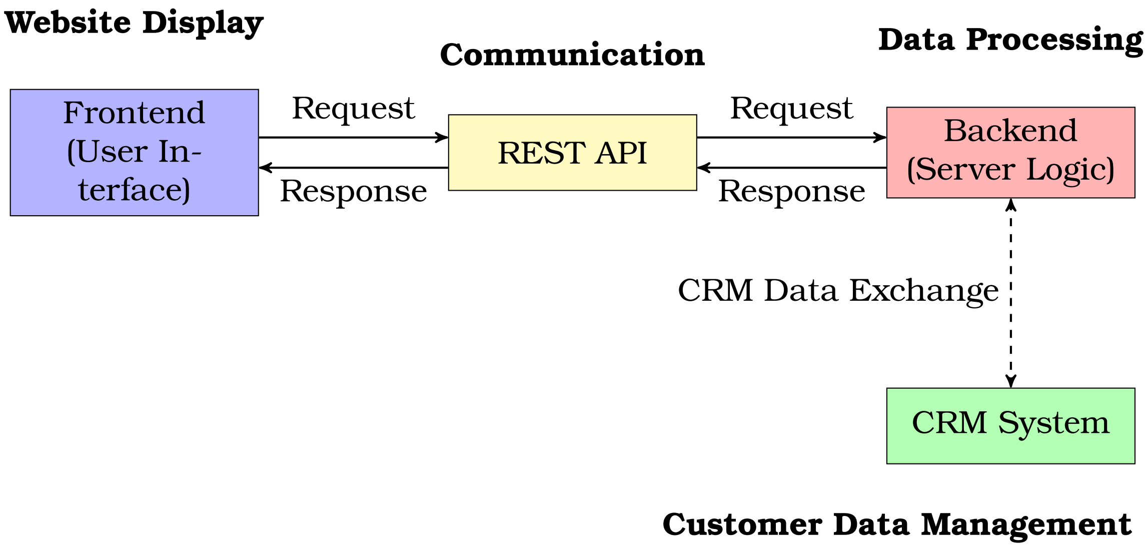
- **Data Warehouses:**
  - **Encryption, Access Control, and Logging**: Encrypt sensitive data, enforce strict access control policies, and maintain detailed logs to monitor data access and modifications.

- **Anonymisation or Pseudonymisation**: Apply these techniques to protect personal data but ensure they are used only when identification is unnecessary for processing.
- **Data Retention Policies**: Define and enforce clear data retention periods.

2 Consent Management: Frontend-Backend Simulation with Real-Time Analytics

2.1 Consent Flow and HTML Implementation

TikZ Scheme



The following HTML (Hypertext Markup Language) script can be integrated into a WordPress website, which includes a frontend (user interface) and a backend (server-side logic, data processing, and functionality supporting the frontend). The frontend sends user information to the backend via a REST API (Representational State Transfer), enabling data exchange using standard HTTP (Hypertext Transfer Protocol) methods without retaining past client states (e.g., session information or login status). If a CRM

- Click **'Manage Preferences'** to customize which cookies you accept.
- You can change your preferences or withdraw consent anytime via our **Cookie Settings**.

2.2 Backend API Programming and Automated Summary Integration

The following Python code sets up a Flask app, serving as a virtual API to simulate communication between the frontend, where users interact with the consent popup, and the backend. The code also initializes a consent database with 50 customer decisions, listens for new inputs, updates cookie preferences when new input is received, and automatically creates a report on customer consents.

Save this code as a .py file in your Downloads folder and run it by double-clicking it. This will open a **command window** that acts as the backend.

Python Code for Virtual API, Backend, and Automated Reporting

```python
import pandas as pd
import numpy as np
from flask import Flask, request, jsonify
import matplotlib.pyplot as plt
import threading  # To handle GUI operations asynchronously
import time  # To add delays if needed
import warnings
import matplotlib

# Ignore unnecessary warnings
warnings.filterwarnings("ignore")

# Set Matplotlib backend
matplotlib.use('TkAgg')  # Ensure interactive backend for GUI operations

# Set a fixed random seed for reproducibility
```

```python
    return jsonify({
        "status": "success",
        "message": f"Consent updated for {user_id}",
        "withdrawal_info": "You can withdraw consent at any time by emailing privacy@yourscience.eu"
    })

if __name__ == "__main__":
    app.run(debug=True, port=5000)
```

Backend Output

```
Microsoft Windows [Version 10.0.19045.5131]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\System32>cd C:/Users/norbert.poncin/Downloads

C:\Users\norbert.poncin\Downloads>python ConsAPIPieSum.py
 * Serving Flask app 'ConsAPIPieSum'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a
production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 626-021-225
```

2.3 Frontend Consent Submission and Backend Update

Run the code in this subsection in **JupyterLab**, which acts as the frontend. Executing the code displays a popup where the customer can select their cookie preferences. These preferences are transmitted to the backend, running in the command window, which then updates the CRM system and prints the complete list of 50+1 customer preferences, including the new entry, as well as a report on Customer Consent Analytics. The backend also sends a confirmation message back to the frontend, indicating the update was successful.

Python Code for Frontend

```python
import requests

# Simulate frontend sending consent decision
def simulate_frontend():
    print("\nWe use cookies to enhance your experience on our website.\n")
    print("Please choose your preferences below:")
    print("- Functional Cookies: Collect essential data for the website to operate (e.g., session IDs). This data is processed locally and not shared.")
    print("- Analytical Cookies: Collect data on website usage (e.g., page views, clicks) to help improve performance. This data may be shared with trusted analytics partners.")
    print("These cookies do not identify you personally or track you across other websites.\n")
    print("You can withdraw your consent at any time by sending an email to privacy@yourscience,eu.\n")


    # Step 1: Input global consent decision
    print("1. Accept All Cookies")
    print("2. Decline All Cookies")
    print("3. Choose Preferences\n")

    global_choice = input("Enter your choice (1/2/3): ").strip()

    # .strip() prevents issues due to whitespace characters like spaces, tabs, or newlines, for instance " 3" instead of "3"

    # Step 2: Define payload based on global choice
    payload = {"userId": "user_051"}

```

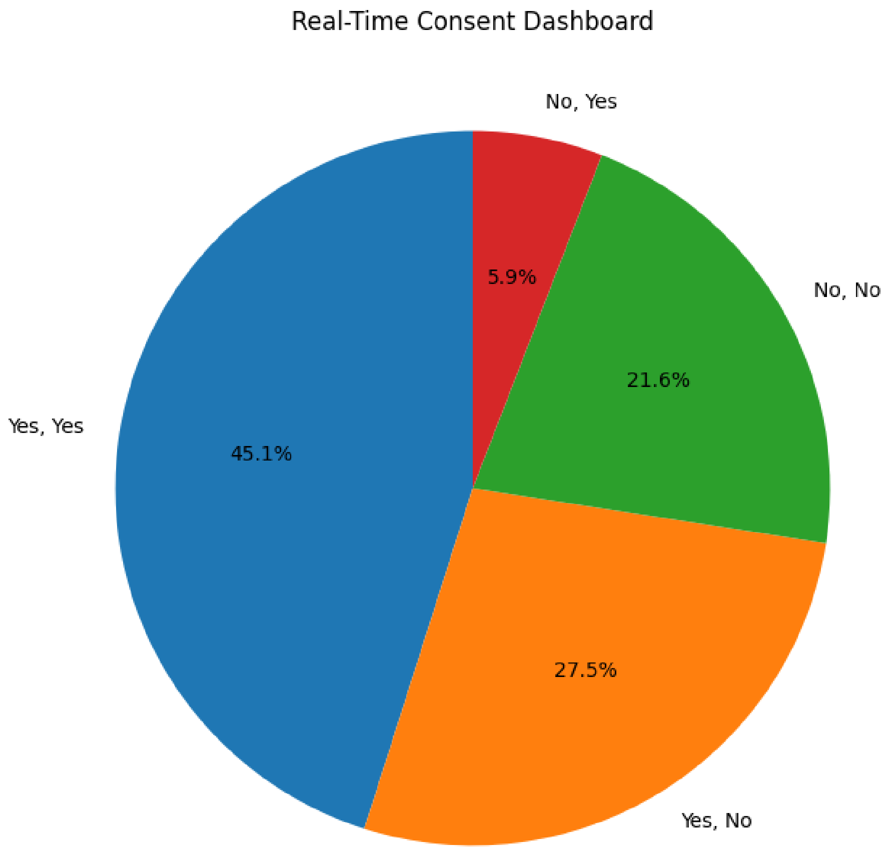


Figure 2: Consent Distribution — Included in the Backend Output

2.4 AI-Generated Comprehensive Insights Report

If your analysis results are shareable, you can upload them to your preferred AI platform and request a professional PDF report. This report can be generated using the ‘consent_db.csv’, the ‘consent_analytics_table.tex’, and the ‘consent_pie_chart.png’, all of which were saved to your Downloads folder by the previous code.

The following report was generated by ChatGPT-ScholarGPT.

Analysis of Customer Consent Data

Norberto Cio

December 31, 2024

Executive Summary

This report presents the findings of an analysis conducted on customer consent preferences for Functional and Analytical Cookies. The purpose of this analysis is to understand user choices and guide the organization's cookie management strategy.

The results indicate that a significant proportion of customers accept both Functional and Analytical Cookies, with smaller groups opting for partial or no consent. Detailed statistics and a visual representation are provided below for further insights.

Findings

The consent preferences have been categorized into four distinct groups:

- **Yes-Yes (Accepted both Functional and Analytical Cookies):**
  Number of Customers: 23
  Percentage of Total: 45.1%

- **Yes-No (Accepted only Functional Cookies):**
  Number of Customers: 14
  Percentage of Total: 27.5%

- **No-Yes (Accepted only Analytical Cookies):**
  Number of Customers: 3
  Percentage of Total: 5.9%

Pages 28–29 are not part of this preview.

Conclusion

This report underscores the need for a balanced approach to cookie manathat respects user preferences while optimizing for business objectives. Please review and provide feedback for further refinement or action plans.

Prepared By: Norberto Cio
Position: CIO
Email: norberto.cio@yourscience.eu

3 Consent Compliance Audit: Self-Paced Activity

Objective: Use Python to simulate an audit of a data warehouse for consent compliance in alignment with GDPR and CCPA regulations.

Scenario: Generate a realistic sample dataset representing a warehouse's stored customer data. The dataset includes:

- **Customer identifiers:** Unique identifiers such as IDs, names, and email addresses.
- **Consent statuses:** Recorded consents for functional, analytical, or marketing purposes.
- **Data usage logs:** Timestamps of actions such as
  - 'email sent' (e.g., marketing emails sent by the company), and
  - 'data shared' (e.g., data shared internally within the company).

Tasks:

- **Consent Validation:** Write Python code to verify if data usage actions comply with the recorded consent statuses. For example, ensure that marketing emails are only sent to customers who have consented to marketing communications.

- **Error Reporting:** Identify and list instances of non-compliance, such as marketing emails sent without the necessary consent.
- **Summarize Compliance:** Generate a report summarizing the compliance rate, such as the percentage of actions compliant with GDPR and CCPA.

Advanced Option: Simulate the revocation of consent:

- Modify the dataset to reflect revoked consents.
- Re-run the compliance checks to account for the changes.

4 Encryption, Anonymization, and Pseudonymization

4.1 Positional Number Systems and Encoding Systems

Number Systems

Below, we briefly discuss the Base 10, Base 2, Base 16, and Base 64 number systems. Base 10 is preferred by humans, while Base 2 suits computers, leveraging a binary architecture: 0 (no current) and 1 (current flows). Bases 16 and 64 encode binary data (e.g., 110110100110) into compact, text-friendly formats (e.g., DA6 and 2m, respectively) for secure transmission over systems like JSON, XML, and email.

- **Decimal Number System** or **Base 10** (**10** coefficients: 0, 1, . . . , 9):

$$2309 = 2 \cdot \mathbf{10}^3 + 3 \cdot \mathbf{10}^2 + 0 \cdot \mathbf{10}^1 + 9 \cdot \mathbf{10}^0 \ .$$

- **Binary Number System** or **Base 2** (**2** coefficients: $0, 1$):

$$\begin{aligned} 86 &= 1 \cdot \mathbf{2}^6 + 0 \cdot \mathbf{2}^5 + 1 \cdot \mathbf{2}^4 + 0 \cdot \mathbf{2}^3 + 1 \cdot \mathbf{2}^2 + 1 \cdot \mathbf{2}^1 + 0 \cdot \mathbf{2}^0 \\ &\simeq 1010110 \simeq 01010110 \ . \end{aligned}$$

The binary digits 0 and 1 are called **bits**, and an 8-bit string is referred to as a **byte**. This gives a concrete meaning to expressions like 5 Megabytes (MB, where 1 MB equals one million bytes) or 3 Gigabytes (GB, where 1 GB equals one billion bytes in the American sense of 10^9).

- **Hexadecimal Number System** (hexa originates from 'héx', meaning 'six') or **Base16** (**16** coefficients):

  The hex digits 0–15 are represented by the decimal digits 0–9 and the letters A–F to ensure single-character representation. Base 16 conveniently encodes 4-bit chunks $0000, 0001, \ldots, 1111$, which correspond to the Base 10 integers 0–15 and align precisely with the hex digits 0–9, A–F:

$$\begin{aligned}
&101101101001011110001101\ldots\ (256\text{-bit Base 2 info}) \quad \rightsquigarrow\\
&\qquad 1011\ 0110\ 1001\ 0111\ 1000\ 1101\ldots\ (64\ \text{4-bit chunks Base 2 info}) \quad \rightsquigarrow\\
&\qquad 11\ 6\ 9\ 7\ 8\ 13\ldots\ (\text{Base 10 info}) \quad \rightsquigarrow\\
&\qquad \text{B6978D}\ldots\ (\text{Base 16 info})\ .
\end{aligned}$$

- **Base 64** (**64** coefficients):

  To ensure single-character representation, the Base 64 digits $0-63$ are represented by the uppercase letters A – Z $(0-25)$, the lowercase letters a – z $(26-51)$, the decimal digits $0-9$ $(52-61)$, and the symbols $+$ (62) and $/$ (63) (the variant URL-safe Base 64 replaces $+$ with $-$ and $/$ with _):

$$\begin{aligned}
&101101101001011110001101\ldots\ (256\text{-bit Base 2 info}) \quad \rightsquigarrow\\
&\qquad 101101101001011110001101\ldots\mathbf{00}\ (258\text{-bit Base 2 info}) \quad \rightsquigarrow\\
&\qquad 101101\ 101001\ 011110\ 001101\ldots\mathbf{00}\ (43\ \text{6-bit chunks Base 2 info}) \quad \rightsquigarrow\\
&\qquad 45\ 41\ 30\ 13\ldots\cdots(\text{Base 10 info}) \quad \rightsquigarrow\\
&\qquad \text{tpeN}\ldots\cdots =\ (\text{Base 64 info})\ .
\end{aligned}$$

Encoding Systems

Number systems and encoding systems work together to facilitate the secure transmission of data.

The most important encoding systems are ASCII and UTF-8:

- **ASCII** (American Standard Code for Information Interchange) encodes the 128 most commonly used symbols in the English language, including letters (A-Z, a-z), decimal digits (0-9), punctuation

  ```
  !"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~
  ```

  and control characters (e.g., the character `LF` (Line Feed) for 'new line').

  Each of the 128 ASCII characters is uniquely represented by a 7-bit binary value from $0 = 0000000$ to $127 = 1111111$, which is often stored in an **8-bit byte** for convenience by adding a leading zero. ASCII is widely used for basic text encoding:

  $$\begin{aligned}\text{`A' is the sixty-fifth of the 128 ASCII symbols} \quad &\rightsquigarrow \\ A = 65 = 1000001 = 01000001 \ .&\end{aligned}$$

- There is an extension of ASCII, referred to as UTF-8, which encodes all Unicode characters. **Unicode** assigns unique codes to characters from almost all writing systems, numerical systems, symbols, and even emojis:

  - **A** (Latin capital letter A): `U+0041`
  - **1** (Digit one): `U+0031`
  - α (Greek small letter alpha): `U+03B1`
  - :-) (Smiling face with open mouth): `U+1F600`

  UTF-8 (Unicode Transformation Format - 8-bit) encodes all Unicode characters using 1, 2, 3, or 4 bytes. It encodes common characters (like ASCII) in just 1 byte (retaining compatibility with ASCII), while supporting complex symbols with 2 to 4 bytes:

Table 3: ASCII Table (0-127) **with** Binary Representation

Dec	Char	Bin	Dec	Char	Bin	Dec	Char	Bin	Dec	Char	Bin
0	NULL	0000000	1	SOH	0000001	2	STX	0000010	3	ETX	0000011
4	EOT	0000100	5	ENQ	0000101	6	ACK	0000110	7	BEL	0000111
8	BS	0001000	9	TAB	0001001	10	LF	0001010	11	VT	0001011
12	FF	0001100	13	CR	0001101	14	SO	0001110	15	SI	0001111
16	DLE	0010000	17	DC1	0010001	18	DC2	0010010	19	DC3	0010011
20	DC4	0010100	21	NAK	0010101	22	SYN	0010110	23	ETB	0010111
24	CAN	0011000	25	EM	0011001	26	SUB	0011010	27	ESC	0011011
28	FS	0011100	29	GS	0011101	30	RS	0011110	31	US	0011111
32	(space)	0100000	33	!	0100001	34	"	0100010	35	#	0100011
36	$	0100100	37	%	0100101	38	&	0100110	39	’	0100111
40	(	0101000	41	)	0101001	42	*	0101010	43	+	0101011
44	,	0101100	45	-	0101101	46	.	0101110	47	/	0101111
48	0	0110000	49	1	0110001	50	2	0110010	51	3	0110011
52	4	0110100	53	5	0110101	54	6	0110110	55	7	0110111
56	8	0111000	57	9	0111001	58	:	0111010	59	;	0111011
60	<	0111100	61	=	0111101	62	>	0111110	63	?	0111111
64	@	1000000	65	A	1000001	66	B	1000010	67	C	1000011
68	D	1000100	69	E	1000101	70	F	1000110	71	G	1000111
72	H	1001000	73	I	1001001	74	J	1001010	75	K	1001011
76	L	1001100	77	M	1001101	78	N	1001110	79	O	1001111
80	P	1010000	81	Q	1010001	82	R	1010010	83	S	1010011
84	T	1010100	85	U	1010101	86	V	1010110	87	W	1010111
88	X	1011000	89	Y	1011001	90	Z	1011010	91	[	1011011
92	\	1011100	93	]	1011101	94	^	1011110	95	_	1011111
96	‘	1100000	97	a	1100001	98	b	1100010	99	c	1100011
100	d	1100100	101	e	1100101	102	f	1100110	103	g	1100111
104	h	1101000	105	i	1101001	106	j	1101010	107	k	1101011
108	l	1101100	109	m	1101101	110	n	1101110	111	o	1101111
112	p	1110000	113	q	1110001	114	r	1110010	115	s	1110011
116	t	1110100	117	u	1110101	118	v	1110110	119	w	1110111
120	x	1111000	121	y	1111001	122	z	1111010	123	{	1111011
124	|	1111100	125	}	1111101	126	~	1111110	127	DEL	1111111

- **A**: `U+0041`, encoded as `01000001` (1 byte)
- **1**: `U+0031`, encoded as `00110001` (1 byte)
- α: `U+03B1`, encoded as `11000011 10110001` (2 bytes)
- :-) : `U+1F600`, encoded as `11110110 10111100 10101111 10011110` (4 bytes)

Since JSON interprets text data in UTF-8 by default, the byte `00001010` (the binary representation of the UTF-8 code `U+000A`, representing the newline character `\n`), for instance, can be misinterpreted during transfer (as a command separator instead of a line separator). This makes raw binary data unsuitable for direct transmission in JSON without encoding – typically in Base64, which prevents such issues by using only safe characters.

Although binary data can cause issues during transfer in certain text-based formats like JSON, we represent text using bytes in ASCII or UTF-8 because computers operate on binary data. Encoding text into bytes provides a standard, efficient way to store, process, and transmit text reliably across binary-compatible systems, such as file systems, network protocols, and databases.

4.2 Hashing, Masking, Tokenization, and Encryption

Main data protection methods include:

- Masking obscures parts of the data (e.g., replacing characters with symbols such as ***). It allows retrieval of the original data and therefore masking is a pseudonymization method (a method where identifiable data like a social security number is replaced with pseudonyms, allowing the data to still be re-identified with access to additional information (e.g., a separate lookup table or key)).

$$\text{3487-6713-3971-4512} \quad \rightsquigarrow \quad \ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast\ast 4512\,.$$

- Tokenization (a ‘token’ is a small object that represents something else, French ‘jeton’, German ‘Marke’, ‘Speilmarke’) replaces sensitive data with unique tokens, with a secure mapping table to reverse the process. Therefore tokenization is a pseudonymization technique.

$$\text{123-456-7890} \quad \rightsquigarrow \quad \text{RDmtrVDX}\,.$$

- Hashing (to hash: French ‘hacher’, German ‘hacken’) processes data in chunks and converts it into a fixed-length string of letters and numbers using a one-way cryptographic function. As it is therefore irreversible, it serves as an example of an anonymization method.

$$\text{mySecurePassword123} \rightsquigarrow \begin{array}{l}\text{ym7lQSBGVTPTZ7TKxc0vEu51}\\ \text{I0IlEw3YlHDeVGq5ykY=}\end{array} .$$

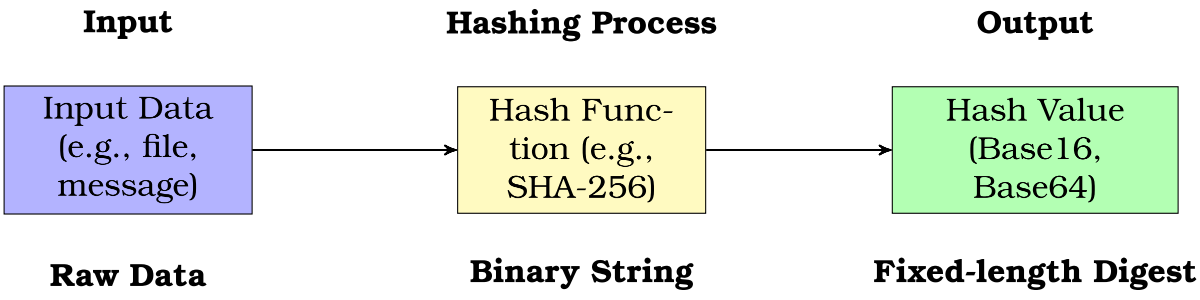

Input
Hashing Process
Output
Input Data (e.g., file, message)
Hash Function (e.g., SHA-256)
Hash Value (Base16, Base64)
Raw Data
Binary String
Fixed-length Digest

Figure 3: Hashing Process

- Encryption transforms data into an unreadable format using an algorithm and a key, which can be reversed through decryption.

$$250.50 \rightsquigarrow \begin{array}{l}\text{gAAAABnP1Apnkj8-FoFcCT1e}\\ \text{XwJDQVI0fubNVckO9Uusl...}\end{array} .$$

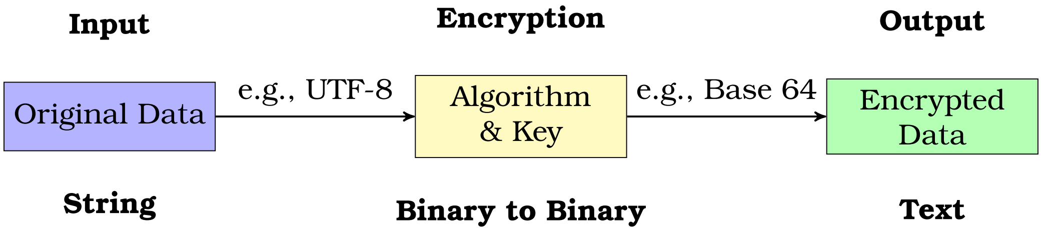


Figure 4: Encryption

The sender encrypts data using a key, and the receiver decrypts it using either the same or a different key (see Figure 5):

- **Symmetric Key Encryption**: Both parties use the same key. The key must be securely shared beforehand.

- **Asymmetric Key Encryption**: The sender encrypts data using the receiver's public key. Only the receiver can decrypt it with their private key. This process often establishes a secure channel to negotiate a symmetric key for ongoing communication.

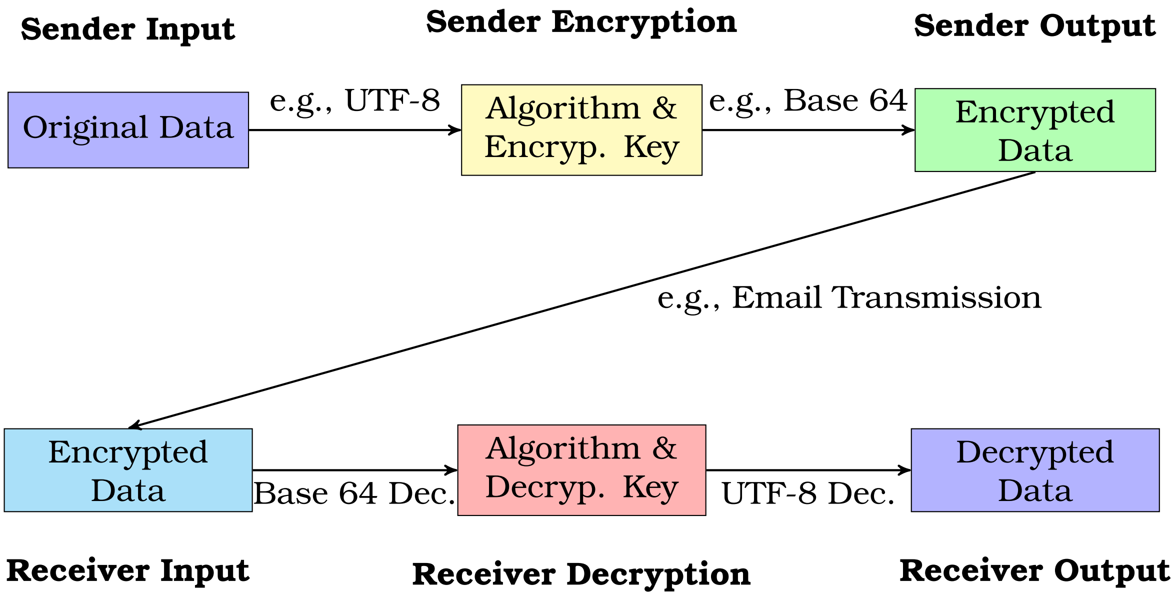


Figure 5: Encryption, Transmission, and Decryption Flow

4.3 Encryption in Warehousing

The subsequent code demonstrates GDPR compliance within a manufacturing company's on-premises data warehouse. **The script functions in the transformation layer**, applying anonymization, pseudonymization, and encryption to safeguard data before it is stored in the storage layer.

Python Code

```python
import pandas as pd
import hashlib
import random
import string
```

```
2  Charlie White  6011-9876-5432-1098  555-123-4567          300.20      P003

  PurchaseDate
0   2023-11-21
1   2023-11-20
2   2023-11-19

Pseudonymized, Anonymized, and Encrypted Data:
  PartnerID PurchaseDate                             CustomerName_Hashed  \
0      P001   2023-11-21  f86206bf359a841e188406e415c195f181ccfff8cd0b98...
1      P002   2023-11-20  7e3d89811312ed290e4d1e50b7edbeea816a31d0b586c5...
2      P003   2023-11-19  f8913a8fc4abe6cacccedb1218a4b7ddaf0667993f1908...

     CreditCard_Masked PhoneNumber_Tokenized  \
0  ***************4512              RDmtrVDX
1  ***************9012              8jyoNpa1
2  ***************1098              T8qeqPhK

                            PurchaseAmount_Encrypted
0  gAAAAABnP1Apnkj8-FoFcCT1eXwJDQVIOfubNVckO9Uusl...
1  gAAAAABnP1ApW-xQwBYQijN9yGOyB5KMudgzhzfTRpN5CQ...
2  gAAAAABnP1ApqlyvTxKVf9oOlL1vp6QLioOlIQGRs4-ZQ3...

Encryption Key (Store Securely):
iB7ju1ODlpXZYFInZ874-8rvbPLHA7z1o2JP8-Kp4y0=
```

Exercise 1. *Analyze the previous Python code to gain a clear understanding of how hashing, masking, tokenization, and encryption are implemented.*

4.4 Secure Broker Communication

The code in this section simulates a data communication system in an industrial environment. It consists of the following key components (see Figure 8):

- **Publishers**: Two on-site Publishers transmit raw, NON-ENCRYPTED data to a Broker within the SAME FACILITY. More precisely, the company uses **MQTT** to efficiently share real-time production statistics and order updates with business partners and managing personnel via the **factory/sensors** and **business/customers** topics, ensuring timely and streamlined collaboration.

- **Broker**: The on-site Broker collects data from the Publishers, *encrypts* it, and transmits it to *remote* Subscribers.

- **Subscribers**: Two remote Subscribers act as monitoring personnel among other roles. One of them undergoes anonymization and pseudonymization to enhance data protection.

- **Hackers**: Two Hackers simulate `security threats` by attempting to intercept data, either from the non-encrypted Broker port or from the encrypted data flow, emphasizing the importance of secure transmission.

To bolster operational security, the system integrates an anomaly detection mechanism using an **Isolation Forest** – an **UNSUPERVISED machine learning model**. This model monitors data streams and triggers warnings when values exceed defined thresholds, ensuring timely detection of production irregularities.

While conceived for educational purposes, this scenario simulates real-world challenges, offering practical exposure to key concepts in industrial data protection essential for **securing modern industrial systems and IoT infrastructures**.

4.4.1 HTTPS over TCP

In HTTPS (Hypertext Transfer Protocol Secure, HTTP secured with TLS) over TCP (Transmission Control Protocol), encryption is automatic and managed through TLS (Transport Layer Security), making this protocol ideal for **request/response** scenarios. In this structure, TLS secures the communication channel, HTTP manages communication logic, including the formatting of headers and body content, handling communication methods (e.g., GET, POST), and the transfer of resources like images, scripts, and HTML (Hypertext Markup Language) documents, TCP ensures reliable data transport (e.g., ordered, lossless delivery), and an API defines interactions (e.g., endpoints such as URLs, data formats) between client and server.

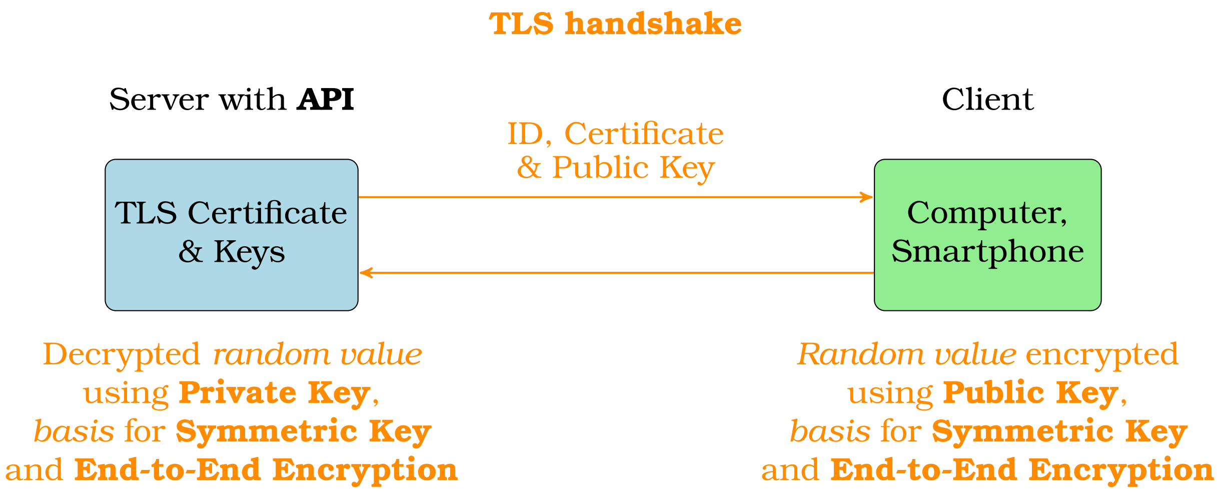

TLS handshake
Server with API
Client
ID, Certificate & Public Key
TLS Certificate & Keys
Computer, Smartphone
Decrypted random value using Private Key, basis for Symmetric Key and End-to-End Encryption
Random value encrypted using Public Key, basis for Symmetric Key and End-to-End Encryption

Figure 7: **HTTPS (HTTP over TLS) on top of TCP**

4.4.2 MQTT over TCP

MQTT (Message Queuing Telemetry Transport) is a lightweight (minimalistic but efficient) protocol designed for real-time communication in challenging environments (e.g., unstable networks). Its name reflects its purpose: ‘Queuing’ for managing delivery with temporary queues when immediate transmission isn’t possible, and ‘Telemetry’ (from Greek ‘tele’ for remote and ‘metry’ for measurement) for data transmission.

In MQTT over TCP (Transmission Control Protocol), encryption must be manually configured through TLS (Transport Layer Security), i.e., MQTT, unlike HTTPS, does not have TLS encryption built-in by default. The MQTT protocol is ideal for **publish/subscribe** scenarios. In this structure, TLS secures the communication channel, MQTT manages communication logic (e.g., topic-based message routing), TCP ensures reliable data transport (e.g., ordered, lossless delivery), and a BROKER facilitates interactions (e.g., managing subscriptions, distributing messages) between publishers and subscribers.

An API, typically *integrated with the backend* on the same server, serves as an interface for interacting with internal processes and clients over HTTPS. In contrast, a BROKER acts as an *independent intermediary*, decoupling publishers and subscribers via MQTT, with TLS *available for secure communication, though not enabled by default.*

4.4.3 Encrypted MQTT

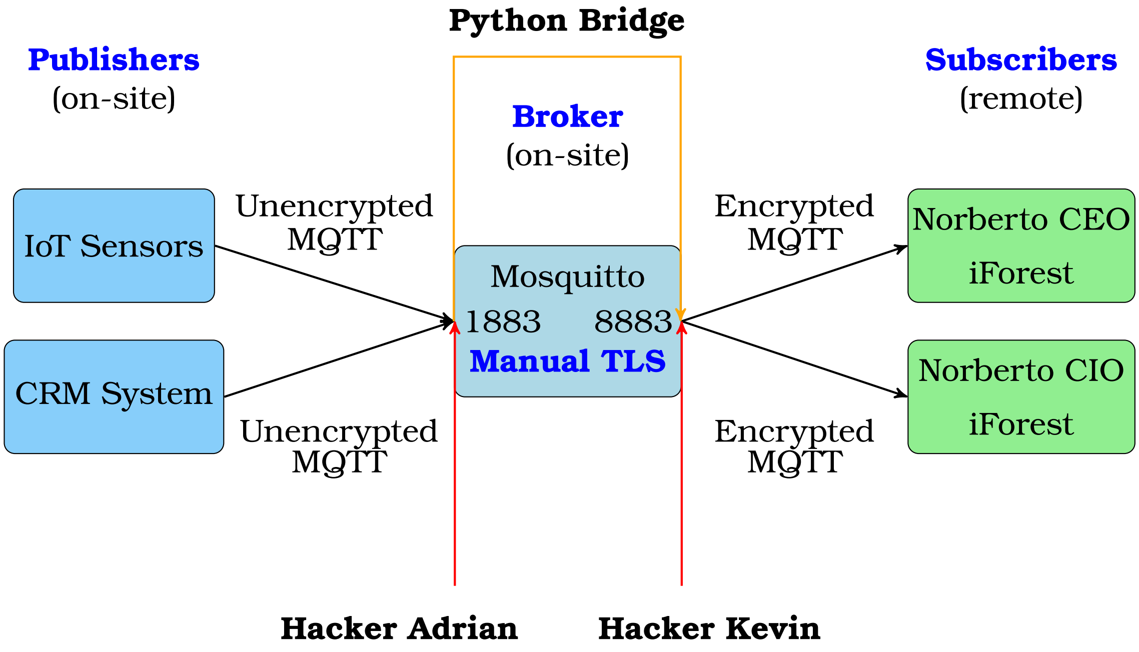


Figure 8: **MQTT Communication Scheme with Bridge and Hacker**

Below are the Python codes that implement the scenario depicted in Figure 8, along with the corresponding outputs. Note that the environment in which these codes are run, JupyterLab or command prompt, and the order of execution significantly impact the results.

4.4.4 Isolation Forests

In an **Isolation Tree**, data points in, for instance, a 2D space defined by features x and y are recursively partitioned by randomly selecting one of these features, x or y, and a random threshold $x = a$ or $y = b$ until each point is fully isolated. The path length required to isolate a point reflects how easily it can be separated: shorter paths indicate potential anomalies, while longer paths suggest denser regions. For example, in a 2D elliptic cloud containing the point $(40, 40)$ at its center, $(22, 45)$ on its left border, and an outlier $(10, 35)$, thresholds like $x = 30$ and $x = 20$ quickly isolate the outlier. However, isolating the central point or other points in denser regions requires finer splits (e.g., $x = 39.5$, $x = 40.5$, and similarly for y), resulting in much longer paths. These **path lengths** are then compared against an expected **threshold** to determine whether a point is an **anomaly**.

An **Isolation Forest** aggregates decisions from multiple trees, with the majority determining whether a data point is an outlier.

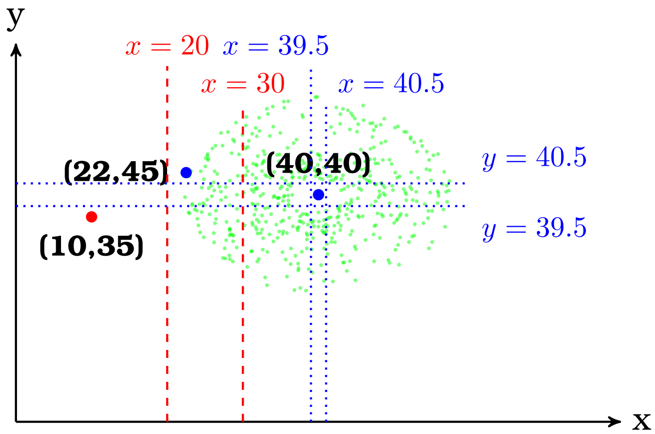


Figure 9: **Isolation Tree**

4.4.5 Python Codes

The Broker

The most practical choice is **Eclipse Mosquitto**, a lightweight MQTT broker widely used in IoT for communication between publishers (e.g., sensors)

```
    if rc == 0:
        logging.info("Listening on port 1883 for unencrypted traffic...")
        for topic, qos in topics:
            client.subscribe(topic, qos)
            logging.info(f"Intercepting topic: {topic}")
    else:
        logging.error(f"Connection failed with code {rc}")

def on_message(client, userdata, message):
    logging.info(f"Snagged message on {message.topic}: {message.payload.decode()}")

# Initialize client and set authentication
client = mqtt.Client(client_id="Adrian")
client.username_pw_set(username, password)  # Send credentials even if not required
client.on_connect = on_connect
client.on_message = on_message

# Connect to the broker and listen
try:
    client.connect(broker_address, port)
    logging.info("Starting MQTT loop...")
    client.loop_forever()
except Exception as e:
    logging.error(f"An error occurred: {e}")
```

4.4.6 Implementation Process

To initiate `factory/sensors` and `business/customers` streams, with unencrypted MQTT communication between IoT devices and operational management systems to the Mosquitto broker, and encrypted communication between the broker and the monitoring subscribers Norberto Ceo and Norberto Cio, follow the steps below.

1. **Start the Mosquitto broker**

   Open a command prompt and navigate to the Downloads directory:

```
C:\WINDOWS\System32>cd C:/Users/norbert.poncin/Downloads
```

Run the Mosquitto configuration file using the command

```
C:/Program Files/mosquitto>mosquitto -c "C:/Users/norbert.poncin/
Downloads/mosquitto.conf" -v
```

Here, `cd` stands for ‘change directory’, the option `-c` specifies the configuration file for Mosquitto to use, and `-v` enables detailed logging. According to the configuration file `mosquitto.conf`, Mosquitto writes its logs to the file `mosquitto.log` in the Downloads folder.

2. **Start the Bridge**

```
C:\WINDOWS\System32>cd C:/Users/norbert.poncin/Downloads
C:/Users/norbert.poncin/Downloads>python Bridge.py
```

Output

```
Python broker bridge running...
```

Once the publisher is running, the bridge receives messages from the broker containing the publisher’s output.

3. **Start the Publisher**

```
C:\WINDOWS\System32>cd C:/Users/norbert.poncin/Downloads
C:/Users/norbert.poncin/Downloads>python Publisher.py
```

Output

```
Press Enter to stop the publisher...
2024-12-01 00:02:09,301 - Publisher connected to Mosquitto broker.
2024-12-01 00:02:09,301 - Published sensor data: {'temperature':
26.39426798457884, 'vibration': 0.12250967970040025, 'timestamp':
1733007729.301026}
2024-12-01 00:02:11,308 - Published sensor data: {'temperature':
22.232107381488227, 'vibration': 0.7628240927476112, 'timestamp':
1733007731.308207}
2024-12-01 00:02:13,308 - Published sensor data: {'temperature':
28.921795677048454, 'vibration': 0.17824494936647456, 'timestamp':
1733007733.3089898}
2024-12-01 00:02:15,309 - Published business data:
{'customer_name': 'Customer 28', 'account_number': 'ACC-343962',
'purchase_amount': 63.40874874713165}
...
```

Explanations

- `2024-12-01 00:02:09,301`: The `301` represents milliseconds, indicating the log was generated at exactly 2 minutes and 9.301 seconds past midnight on December 1, 2024.
- `'vibration': 0.12250967970040025`: Vibrations are scaled to the range $[0, 1]$ for easier comparison.
- `'timestamp': 1733007729.301026`: Represents the number of seconds elapsed since January 1, 1970, 00:00:00 UTC (Unix Time), not counting leap seconds (additional seconds added to UTC to align atomic time with Earth's slightly irregular rotation). UTC is the modern successor to GMT (Greenwich Mean Time).

4. **Start the first Subscriber**

```
C:\WINDOWS\System32>cd C:/Users/norbert.poncin/Downloads
C:/Users/norbert.poncin/Download>python Subscriber.py
```

Output

```
Enter your username: Norberto Ceo
Enter your password: noceBoss
2024-12-01 00:53:02,428 - Starting MQTT loop...
2024-12-01 00:53:02,428 - Connected to broker successfully.
2024-12-01 00:53:02,428 - Subscribed to topic: factory/sensors
2024-12-01 00:53:02,428 - Subscribed to topic: business/customers
2024-12-01 00:53:11,716 - Received sensor data: {'temperature':
21.024178580781534, 'vibration': 0.23909942323747727, 'timestamp':
1733010791.4976332}
2024-12-01 00:53:13,593 - Received sensor data: {'temperature':
21.88416784965297, 'vibration': 0.9073510594873851, 'timestamp':
1733010793.4995441}
2024-12-01 00:53:15,594 - Received sensor data: {'temperature':
28.314161818660438, 'vibration': 0.5730875422169475, 'timestamp':
1733010795.500364}
2024-12-01 00:53:17,611 - Business data: {'customer_name':
'Customer 97', 'account_number': 'ACC-941566', 'purchase_amount':
298.0007727892637}
2024-12-01 00:53:17,709 - Received sensor data: {'temperature':
90.94223720253711, 'vibration': 9.617048763386137, 'timestamp':
1733010797.5020158}
2024-12-01 00:53:21,615 - Received sensor data: {'temperature':
28.170387032772446, 'vibration': 0.9007537468435135, 'timestamp':
1733010801.5056877}
2024-12-01 00:53:23,673 - Business data: {'customer_name':
'Customer 44', 'account_number': 'ACC-280572', 'purchase_amount':
393.6152956377726}
```

```
...
2024-12-01 00:53:51,643 - Received sensor data: {'temperature':
26.981329067750316, 'vibration': 0.2666132010199073, 'timestamp':
1733010831.5343747}
2024-12-01 00:53:51,690 - Processing sensor data for anomalies...
2024-12-01 00:53:51,922 - Anomalies detected: [[90.94223720253711,
9.617048763386137], [90.94223720253711, 9.617048763386137],
[55.9461108999436, 6.265062356117424], [55.9461108999436,
6.265062356117424]]
```

5. **Start Hacker Kevin**

```
C:\WINDOWS\System32>cd C:/Users/norbert.poncin/Downloads
C:\Users\norbert.poncin\Downloads>pip install scapy numpy scikit-learn
C:\Users\norbert.poncin\Downloads>python HackerKevin.py
```

Output

```
Listening on port 8883 for encrypted traffic...
Captured Packet:
Payload (Hex): b'170303001a0000000000000008245f15fc9984db15fe56e7e
7217dfee77771'
Payload (ASCII): $_§§V!}wq
Payload is encrypted.
No raw payload in the packet.
Captured Packet:
Payload (Hex): b'170303001a00000000000000095110c8d10b3c1e52e02c62
b4055cc020997c'
Payload (ASCII): §¬!●→   Q<R,b\
Payload is encrypted.
```

```
No raw payload in the packet.
Captured Packet:
Payload (Hex): b'170303001a000000000000000ac04d1c7bd44c43b1619976
e46f5c49a30b65'
Payload (ASCII): !?→M{LCavo\Ie
Payload is encrypted.
No raw payload in the packet.
```

You have to run the second subscriber before getting results.

6. **Start the second Subscriber**

```
C:\WINDOWS\System32>cd C:/Users/norbert.poncin/Downloads
C:\Users\norbert.poncin\Downloads>python Subscriber.py
```

Output

```
Enter your username: Norberto Cio
Enter your password: nociboss
2024-12-02 20:13:53,469 - Starting MQTT loop...
2024-12-02 20:13:53,469 - Connected to broker successfully.
2024-12-02 20:13:53,469 - Subscribed to topic: factory/sensors
2024-12-02 20:13:53,469 - Subscribed to topic: business/customers
2024-12-02 20:13:53,645 - Received sensor data: {'temperature':
24.777291872656875, 'vibration': 0.8399040807694873, 'timestamp':
1733166833.5358663}
2024-12-02 20:13:55,661 - Received sensor data: {'temperature':
57.45987922368276, 'vibration': 7.814197985422821, 'timestamp':
1733166835.5369482}
2024-12-02 20:13:57,647 - Anonymized business data: {'customer_id':
'ANONYMIZED', 'purchase_amount': 186.72598032900245}
```

```
1733166837.5379777}
2024-12-02 20:13:57,662 - Received sensor data: {'temperature':
25.580730346176285, 'vibration': 0.4745566192692956, 'timestamp':
2024-12-02 20:14:03,637 - Anonymized business data: {'customer_id':
'ANONYMIZED', 'purchase_amount': 242.9734262631916}
2024-12-02 20:14:15,742 - Received sensor data: {'temperature':
63.28162358203201, 'vibration': 5.711181427218387, 'timestamp':
1733166855.5547125}
2024-12-02 20:14:49,677 - Received sensor data: {'temperature':
98.65734079108472, 'vibration': 8.044358335409825, 'timestamp':
1733166889.5997086}
...
2024-12-02 20:15:31,755 - Processing sensor data for anomalies...
2024-12-02 20:15:31,911 - Anomalies detected: [[63.28162358203201,
5.711181427218387], [98.65734079108472, 8.044358335409825],
[99.61285696976098, 9.56464939242084], [57.207770798347376,
9.354664856746098], [81.55805084273203, 9.604645493901483]]
```

7. **Start Hacker Adrian**

```
C:\WINDOWS\System32>cd C:/Users/norbert.poncin/Downloads
C:\Users\norbert.poncin\Downloads>python HackerAdrian.py
```

Output

```
Listening on port 1883 for unencrypted traffic...
2024-12-02 20:13:53,469 - Intercepting topic: factory/sensors
2024-12-02 20:13:53,469 - Intercepting topic: business/customers
2024-12-02 20:14:03,637 - Snagged message on business/customers:
{'customer_name': 'Customer 97', 'account_number': 'ACC-941566',
'purchase_amount': 298.00077278926376}
2024-12-02 20:14:15,742 - Snagged message on factory/sensors:
{'temperature': 63.28162358203201, 'vibration':
```

```
5.711181427218387, 'timestamp': 1733166855.5547125}
2024-12-02 20:14:49,677 - Snagged message on factory/sensors:
{'temperature': 98.65734079108472, 'vibration': 8.044358335409825,
'timestamp': 1733166889.5997086}
...
```

5 Learning Outcomes

After working through this chapter, the reader will have developed practical expertise in **Data Security, Industrial Automation, and Compliance**, while leveraging modern **AI and Encryption Techniques**. In particular, they should be able to:

- **Ensure GDPR Compliance**: Understand and implement key data protection measures to comply with the EU's General Data Protection Regulation (GDPR).
- **Secure Frontend-Backend Communication**: Describe and implement API-based HTTPS communication, ensuring GDPR compliance in Python-based industrial applications.
- **Automate Data Analysis & Reporting**: Develop automated data analysis workflows within a CRM system to generate official reports.
- **Master Data Encoding & Anonymization**: Apply Base64 and UTF-8 encoding techniques and implement data 'anonymization' methods, including masking, tokenization, hashing, and encryption.
- **Implement Industrial Cybersecurity**: Strengthen data protection in industrial systems and IoT infrastructure to prevent security breaches.
- **Differentiate Secure Communication Protocols**: Compare HTTPS (with API-based encryption) and MQTT (with TLS encryption via an independent broker), including a practical understanding of the TLS handshake.
- **Apply AI for Industrial Anomaly Detection**: Understand the Isolation Forest algorithm and use it to detect anomalies in manufacturing environments.
- **Simulate Real-World Industrial Systems**: Model APIs, CRMs, Frontend popups, secure credentials, encrypted an partially anonymized real-time data flows, and other Python-based industrial scenarios.

YOUR SCIENCE

CREATING OPPORTUNITY

About the Author

Norbert Poncin is a Luxembourgish mathematician, who was originally educated as a mathematical analyst and has worked extensively in partial differential equations (PDEs) at the University of Liège. His Master's thesis focused on the propagation of singularities in boundary value problems (BVPs) for dynamic hyperbolic systems. Applying the finite element method (FEM), his subsequent dissertation addressed BVPs for complex elliptic systems of PDEs. For his doctoral thesis, he explored mathematical quantization, while his post-doctoral education at the Polish Academy of Sciences strongly emphasized theoretical physics and its models.

Norbert has served as a Full Professor of Mathematics at the University of Luxembourg for more than 25 years and collaborated with more than 25 foreign professors and post-doctoral scholars. He has organized numerous academic events, notably approximately 10 international research meetings and over 20 research seminars focusing on theories, frameworks, concepts and models in Physics and Engineering. Beyond a substantial publication record in Differential Geometry, Algebraic Topology, and related disciplines, he has contributed roughly 25 papers to the fields of Mathematical Physics and Quantum Theory.

He was the leading instructor for over 20 university courses. Spanning a diverse spectrum of subjects, including mathematical analysis, probability theory, inferential statistics, point and solid dynamics, Lagrangian and Hamiltonian mechanics, mechanics of deformable solids, fluid dynamics, special relativity, quantum physics, geometric methods in mathematical physics, and supersymmetric models, his teaching portfolio underscores his extensive experience in applied aspects of mathematics.

In 2023, Norbert Poncin founded the mathematical consulting agency Your Science, where he currently serves as director. His primary interests include data science and artificial intelligence, along with mathematical modeling and computational science.