Watermarking
(Defense and Hazards)

Luca Caviglione
Institute for Applied Mathematics and Information Technologies
National Research Council of Italy
luca.caviglione@ge.imati.cnr.it

Summer School on Artificial Intelligence for a Secure Society
5 - 10 September 2024
Capo Vaticano Resort Thalasso SPA
Capo Vaticano, Calabria, Italy

AI 4 SECURE SOCIETY

SOBIGDATA
RESEARCH INFRASTRUCTURE

Consiglio Nazionale
delle Ricerche

Course Goal and Organization

- This is a **crash course** on **watermarking**
- Spoilers:
  - techniques are tightly-coupled with the application
  - emphasis on the AI world
  - watermarking is a double-edged sword!

- Organization
  - Day 1 (9 September – 1h): basic information, the big picture, and examples
  - Day 2 (10 September – 2h): more examples, security, and challenges
- All the needed material is available on the course GitHub:
  - https://github.com/lucacav/AI4SS

Motivations (1)

- **Explosion** in the diffusion of **digital information**
- Some examples:
  - **digital media**: audio, video, pictures, and text
  - **software**: source code, binaries, and libraries
  - **3D mesh**: entertainment, lidar scans, 3D-printed artifacts
  - **urban intelligence applications**: IoT data, maps, and measurements
  - **medical data**: imaging and ad-hoc protheses
  - **network traffic**: routing, QoS, and traffic engineering
  - **hardware**: FPGA and custom silicon
  - **artificial intelligence**: datasets, models, and outputs

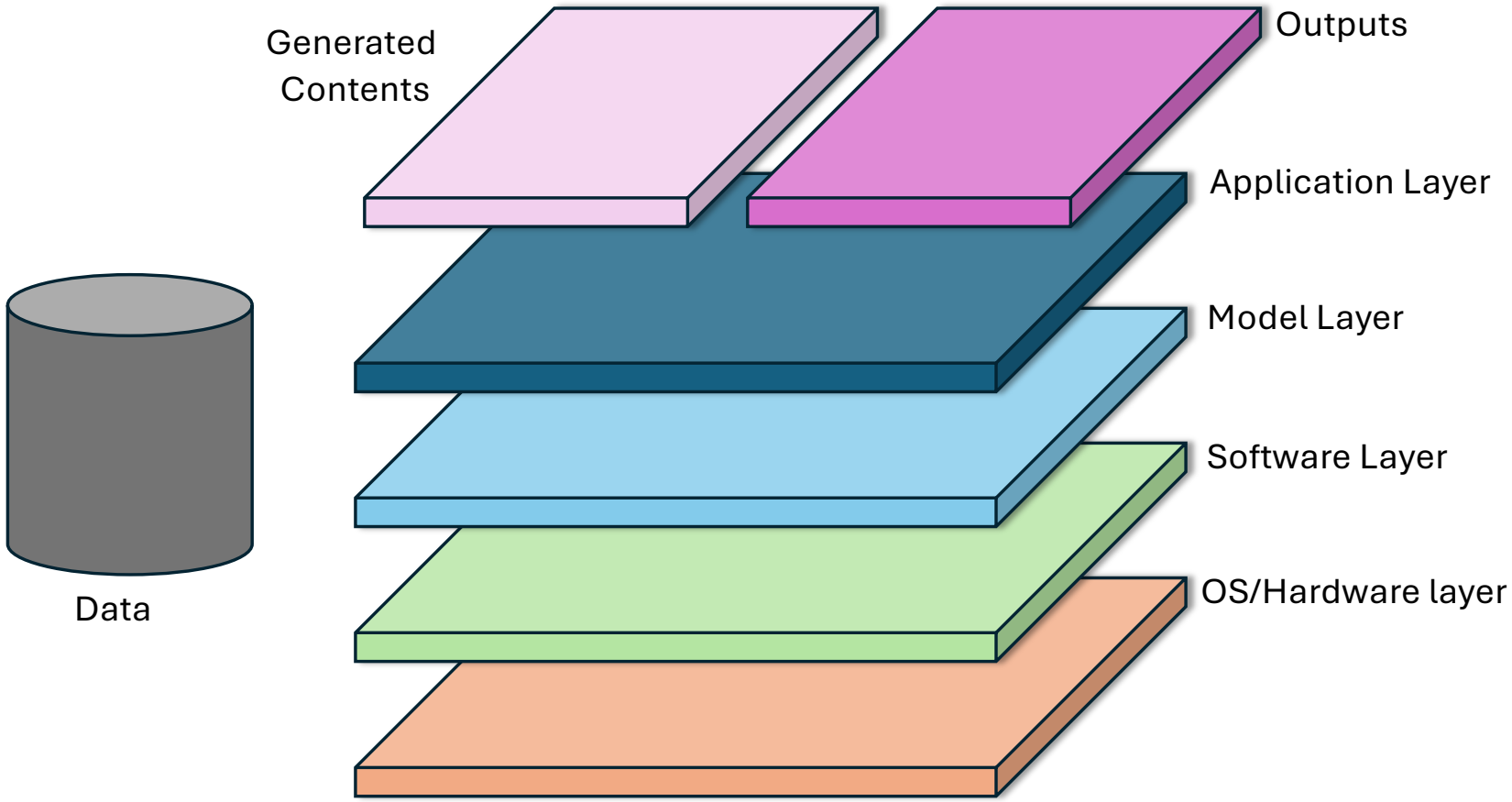
Motivations (2)

- Technological, ethical, and economic **drivers**:
  - diffusion of the “as-a-Service” paradigm
  - industrial competition
  - digital sovereignty
  - protection of the value
  - misinformation and manipulation
- **Digital information** requires (efficient) **mechanisms** for:
  - copyright protection
  - fingerprinting and tracking
  - integrity
  - authentication
  - annotation/metadata

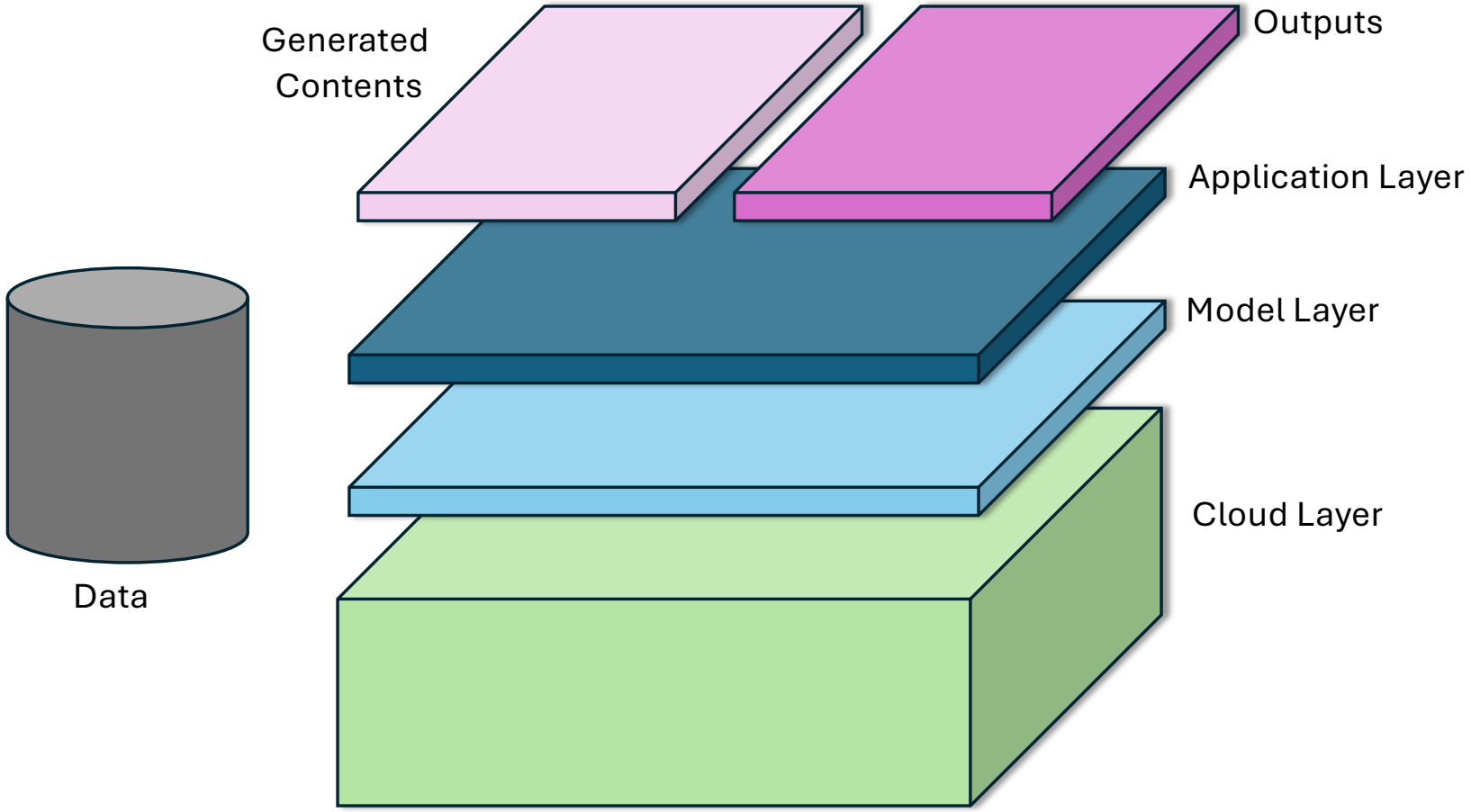
Watermarking and AI (1)

- **Watermarking** is about **embedding** information into original data to:
  - claim something, e.g., copyright/ownership
  - not affect the usability
- AI is **not** only about AI:
  - datasets and (pre-trained) models
  - software libraries and running code
  - dedicated hardware
  - generated contents and as-a-Service frameworks
- There are many **issues** that (still) require a **solution**:
  - detect poisoning
  - enforce guarantees in ML-as-a-Service
  - copyright
  - …

Watermarking and AI (2)



Watermarking and AI (2)



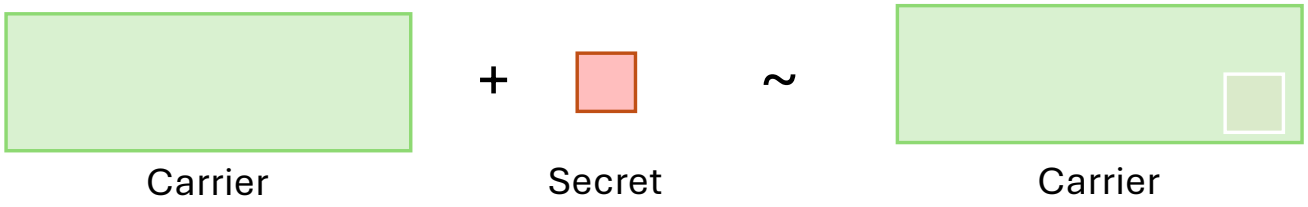
Watermarking and AI (2)

There is no a one-size-fits-all mechanism for watermarking all the entities composing an AI stack!

Basic (1)

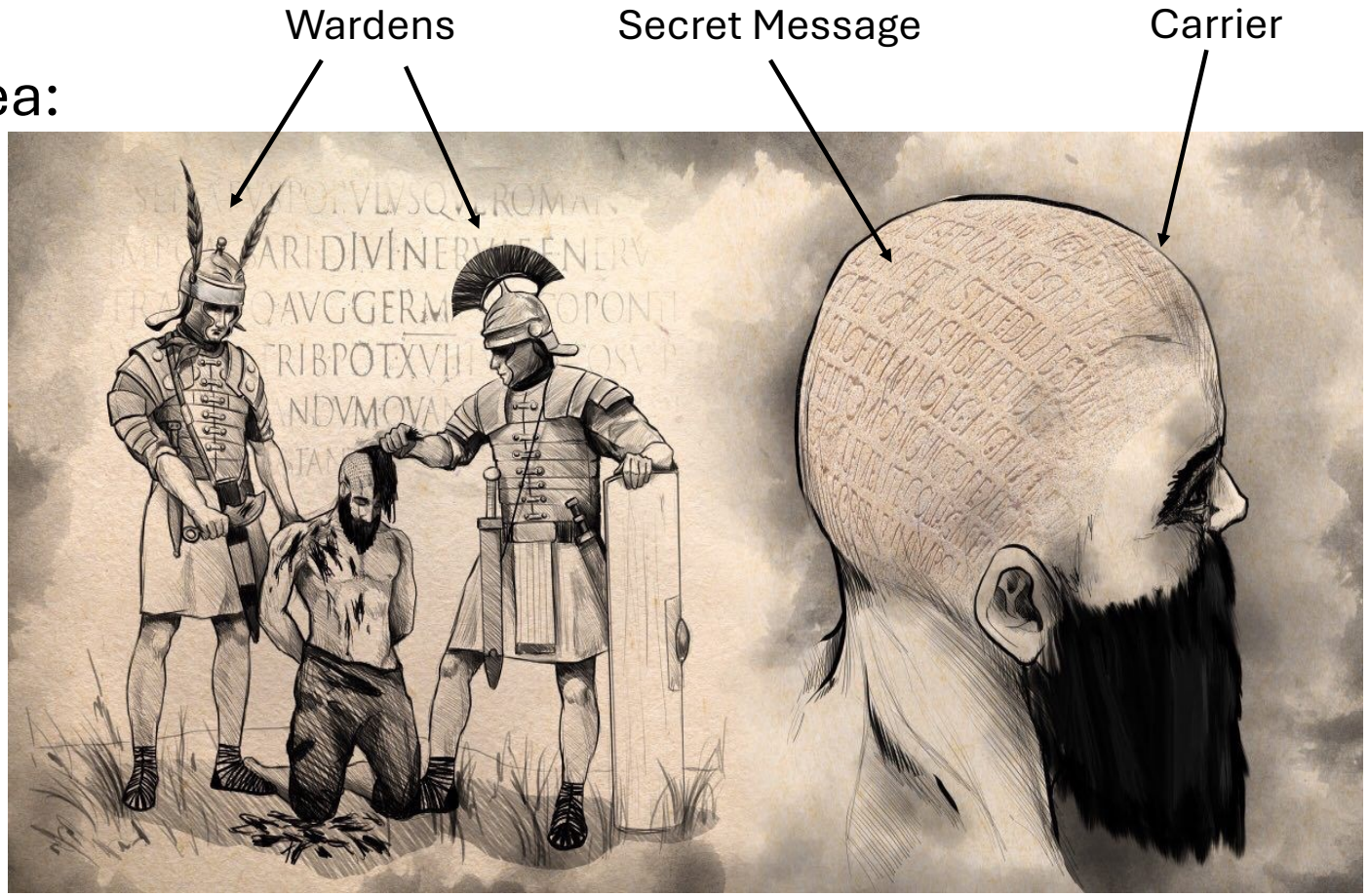
- The word **steganography** is the combination of:
  - *steganos* (στεγᾰνός) = covered or concealed
  - *graphe* (γραφή) = writing
- The **first** recorded use of the term:
  - in 1499 by Johannes Trithemius
  - book “Steganographia”, i.e., an essay on cryptography and steganography
- Mentioned in 440 BC by Herodotus in his Histories
- Cloak secret data into a suitable carrier

Johannes Trithemius



Basic (2)

- Key idea:



Source: https://medium.com/@z3roTrust/using-digital-steganography-to-protect-national-security-information-463bba664830

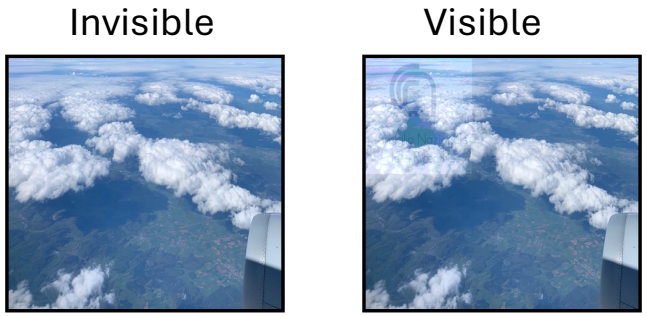
Basic (3)

- Watermarking mechanisms have many **properties/requirements**
- Basic:
  - **capacity**: how much information can be embedded
  - **robustness**: how the data can withstand (un)wanted removal attempts
  - **secrecy**: how difficult is to spot the secret information
- Other(*):
  - **fidelity**: how much degradation is caused to the carrier
  - **reliability**: owner can identify secrets with a high probability, i.e., low F_N
  - **integrity**: owner rarely accuses another owner, i.e., low false alarm
  - **efficiency**: the price paid for embedding/verifying the watermark
  - **generality**: is the mechanism dependent on a specific carrier?

(*) F. Boenisch, “A Systematic Review on Model Watermarking for Neural Networks”, Frontiers in Big Data, Vol. 4, pp. 4:1 – 4:16, Nov. 2021

Basic (4)

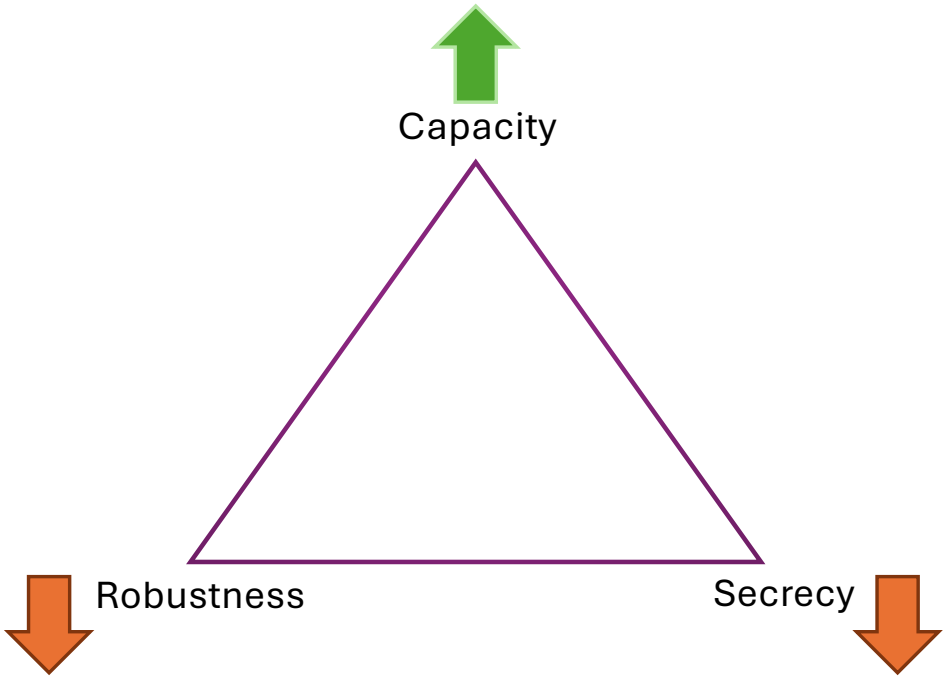
- Another possible characterization:
  - **visible**
  - **invisible**



- The ultimate **goal** of the watermark also plays a role:
  - **robust**: suitable if the carrier is expected to be processed multiple times (e.g., transcoding) or attacked by a malicious actor
  - **fragile:** suitable in case minimal alterations of the carrier need to be revealed or tracked

Basic (5)

- Two “pillars” of **information hiding** are(*):
  - the needle in the haystack principle
  - the magic triangle



(*) W. Mazurczyk, L. Caviglione, “Steganography in Modern Smartphones and Mitigation Techniques”, IEEE Communications Surveys & Tutorials, Vol. 17, No. 1, pp. 334-357, First Quarter 2015

Examples of Watermarking Techniques

Roadmap

- The various watermarking techniques will be explored via **examples**:
  - digital media
  - 3D mesh
  - network traffic
  - software
  - hardware
  - AI models and outputs
- Choice:
  - showing a mechanism from **different points of view**
  - emphasizing the **hiding approach - carrier** relationship
  - have some nice **pointers** for further investigation
  - have fun!

Digital Media (1)

- Digital media are **ubiquitous**:
  - medical
  - entertainment
  - applications assets, e.g., icons and artworks
  - …
- The literature offers a **huge** variety of watermarking methods:
  - each one tailored for a specific goal or carrier
- **Two** main categories:
  - spatial domain
  - frequency domain (e.g., DCT and DFT)

Digital Media (2)

- Least Significant Bit (LSB) watermarking is the most **famous** family of methods:
  - easy yet effective
  - very popular and portable
- Directly rooted from LSB steganography, it can be used with:
  - digital pictures
  - digital video/audio
  - timestamps
  - numerical values
  - …

Not only for digital media!
More later!

Digital Media (2)

- LSB watermark requires the carrier to **resist** a single-bit modification:
  - not suitable for software or “structured data”!

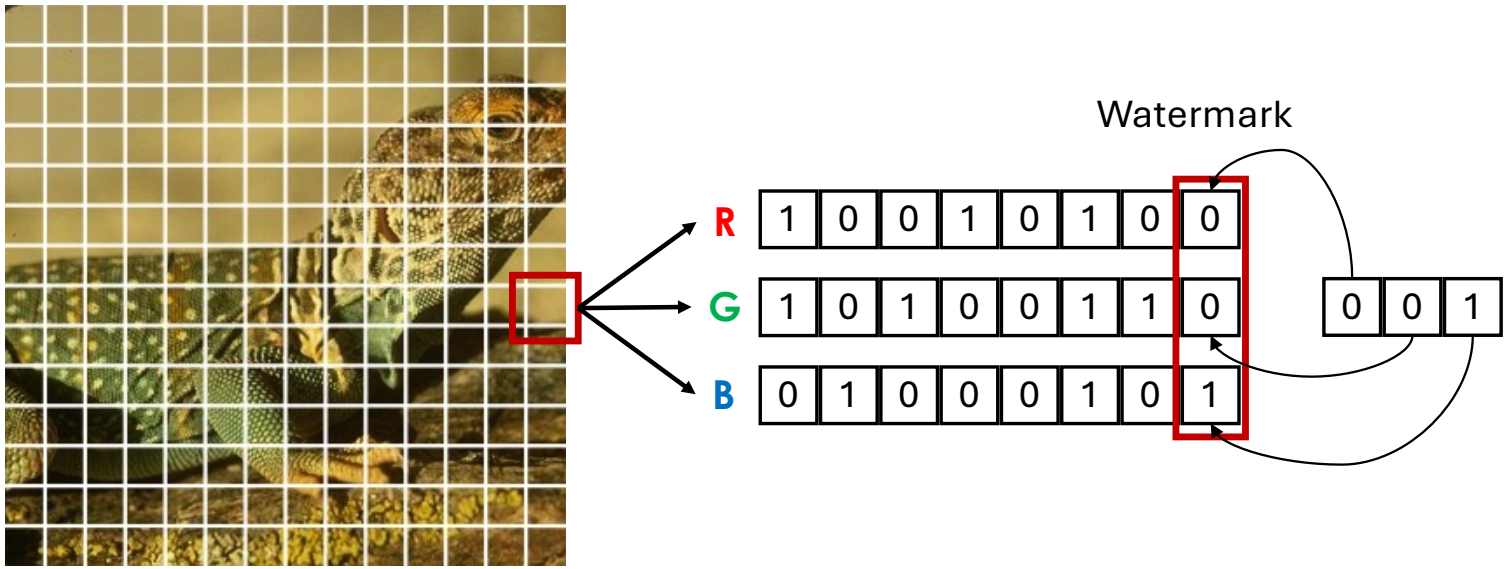
Hex Dump of .pdf

Altering a bit will mostly render the document unreadable
(not to mention that it is easy to spot by computing the MD5)

LSB Watermarking (1)

- As a paradigmatic example, will consider digital pictures

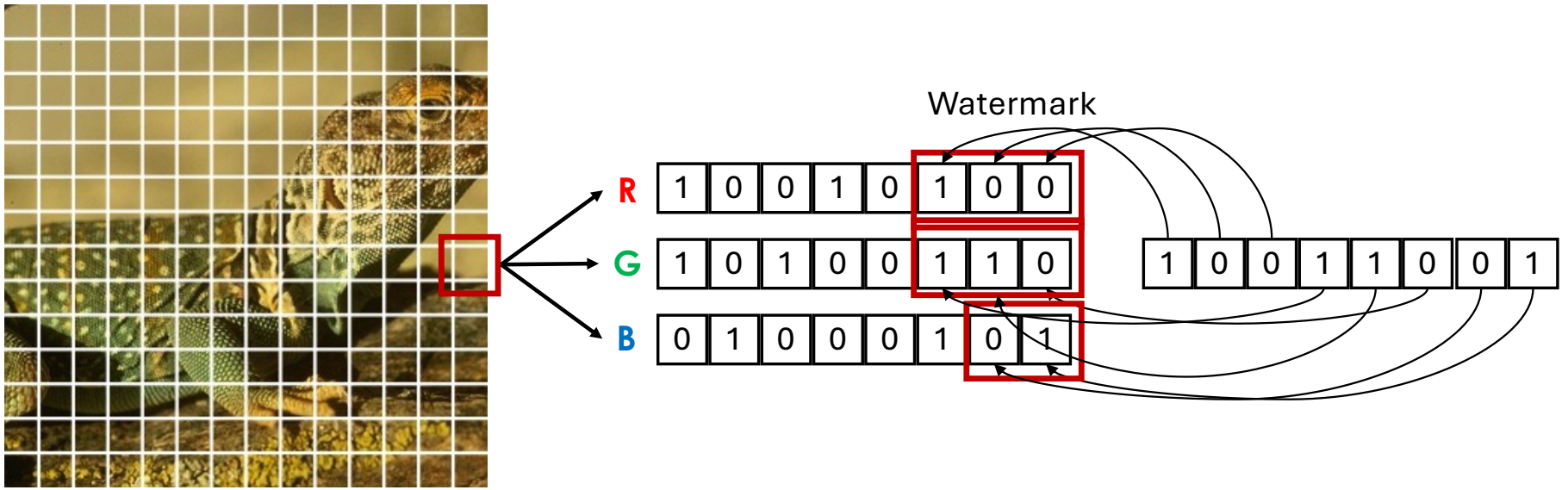
LSB “standard”



LSB Watermarking (1)

- As a paradigmatic example, will consider digital pictures

LSB Multi-Bit



LSB Watermarking (2)

- Demo (1 Bit – Red Channel):

Original Image

Watermarked Image

Secret: 100 times “Summer School on Artificial Intelligence for a Secure Society”

LSB Watermarking (2)

- Demo (1 Bit – Red Channel):

Blue Channel
(Grayscale)

Original Image

Watermarked Image

LSB Watermarking (2)

- Demo (1 Bit – Red Channel):

Green Channel
(Grayscale)

Original Image

Watermarked Image

LSB Watermarking (2)

- Demo (1 Bit – Red Channel):

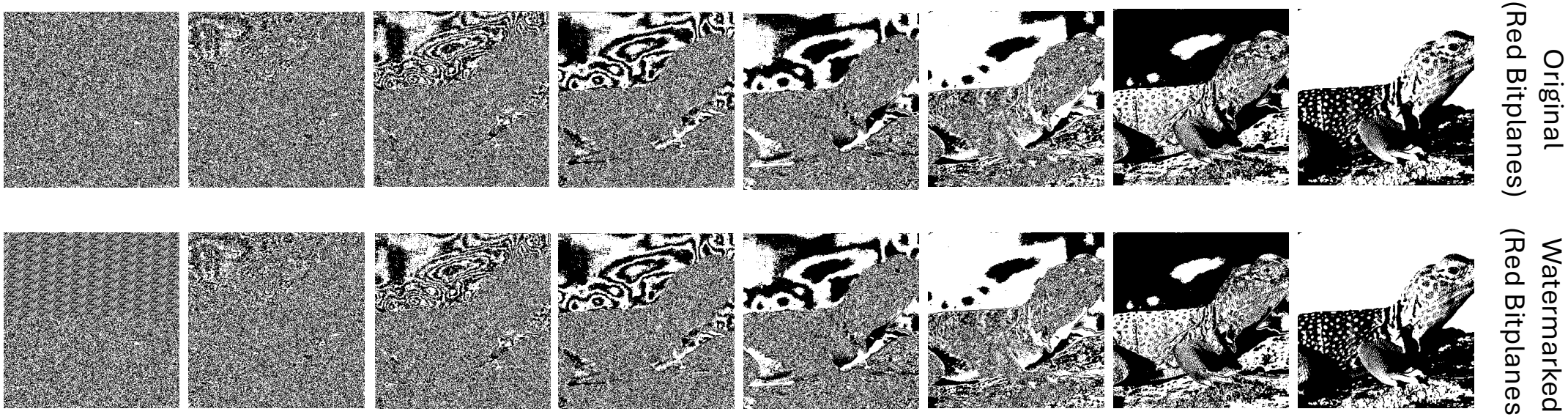
Red Channel
(Grayscale)

Original Image

Watermarked Image

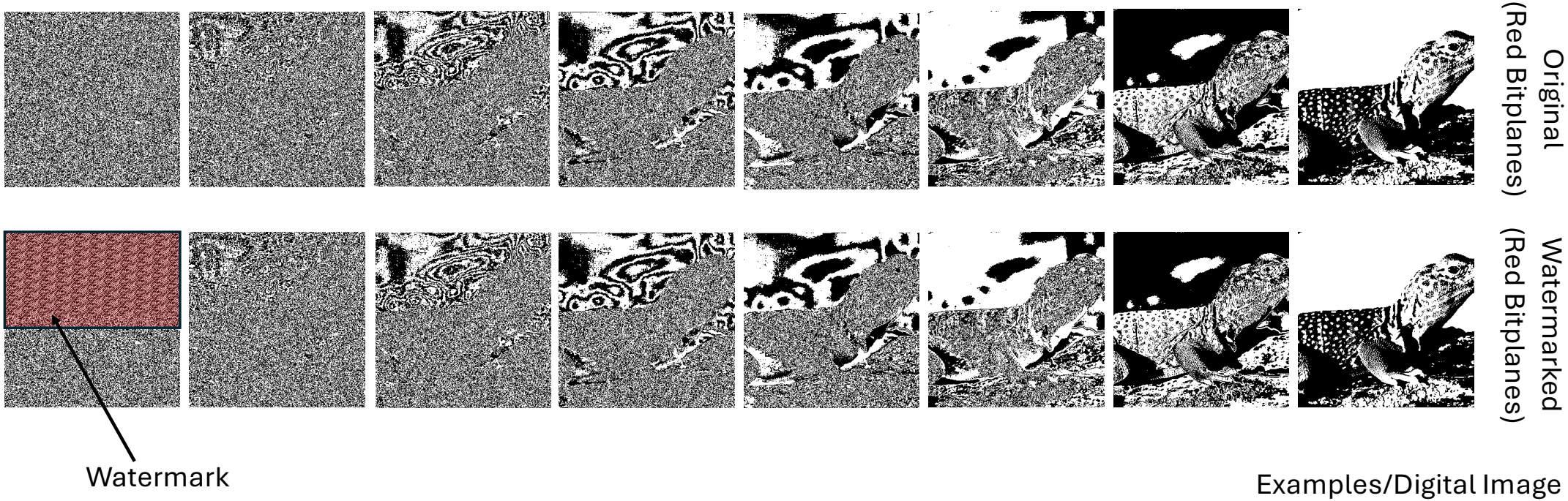
LSB Watermarking (2)

- Demo (1 Bit – Red Channel):



LSB Watermarking (2)

- Demo (1 Bit – Red Channel):



3D Mesh (1)

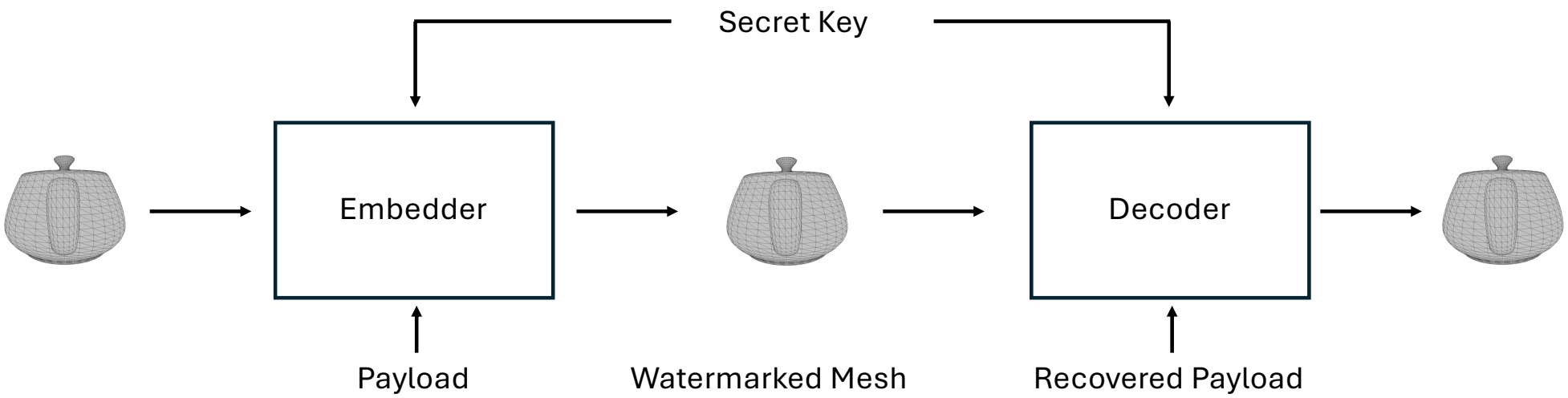
- 3D meshes are at the **basis** of many emerging applications:
  - virtual reality
  - video games / entertainment
  - medical imaging
  - CAD
  - printing
- A **mesh** is composed of:
  - vertices, edges, and facets
  - geometry (3D positions via coordinates)
  - connectivity (adjacency relations among elements)
- More complex data requires a more complex approach

3D Mesh (2)

- The most valuable meshes are:
  - **difficult** to obtain (e.g., require the work of several artists)
  - very **valuable** (obtained through expensive data gathering/scanning campaigns)
- Literature proposes various watermarking techniques
- **Fragile** methods:
  - typically used for authentication, should disrupt against minimal modifications
  - should locate or identify the attack
  - can hide data in spatial (connectivity) or transformed (spectral) domains
- **Robust** methods:
  - typically used for IP management, should resist against great distortions/degradations
  - should balance robustness and secrecy
  - can hide data in spatial (connectivity) or transformed (spectral) domains

3D Triangle Mesh (1)

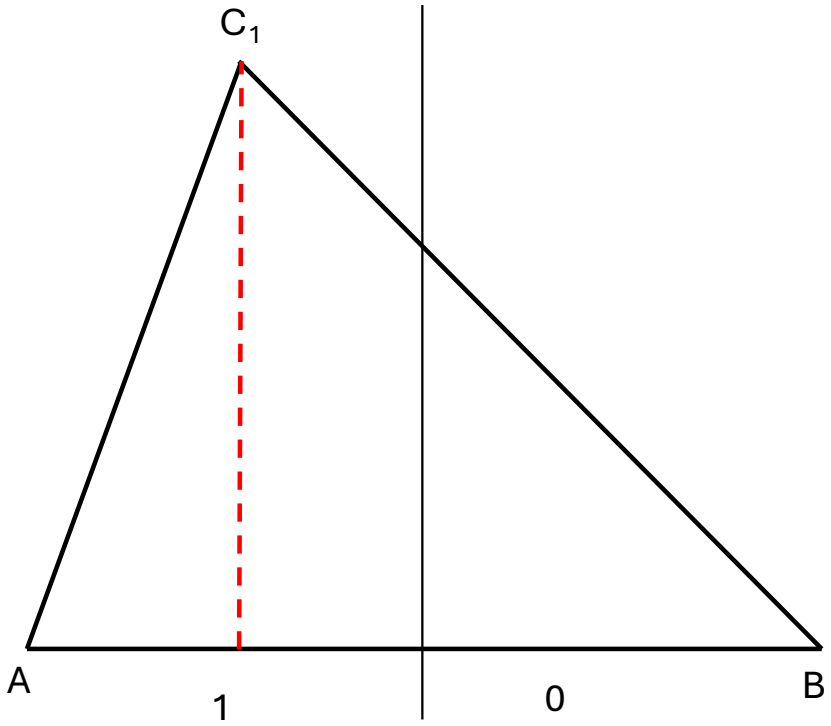
- Example of a **fragile** technique in the **spatial** domain:
  - based on the modification of the geometry
  - for triangle meshes
  - method by Cayre and Macq (*)



(*) F. Cayre, B. Macq, “Data hiding on 3-D triangle meshes”, IEEE Transactions on Signal Processing, Vol. 51, No. 4, pp. 939-949, April 2003

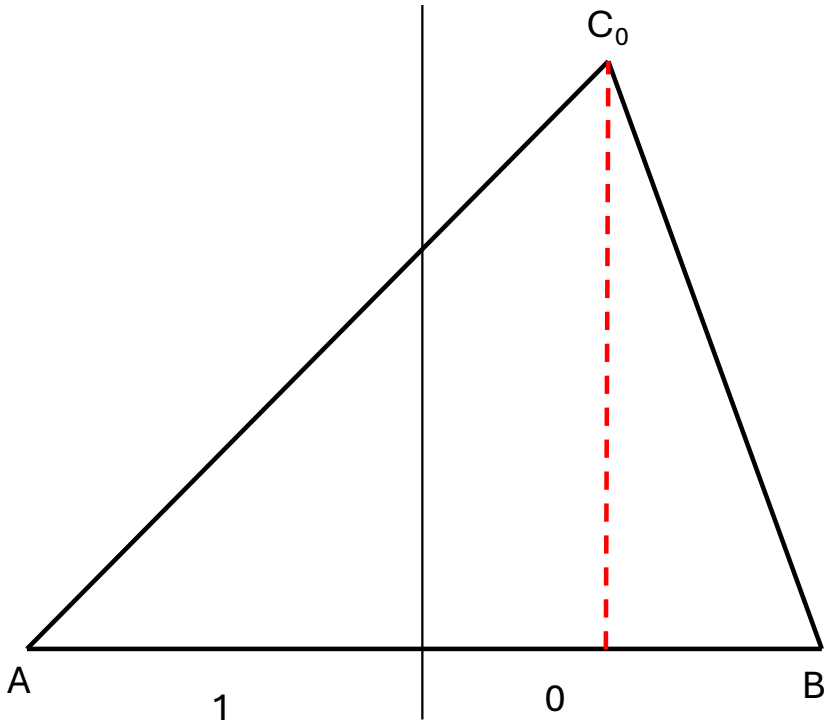
3D Triangle Mesh (2)

- Key idea:



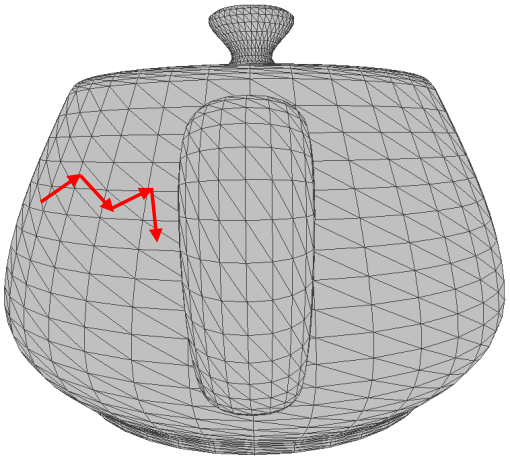
3D Triangle Mesh (2)

- Key idea:

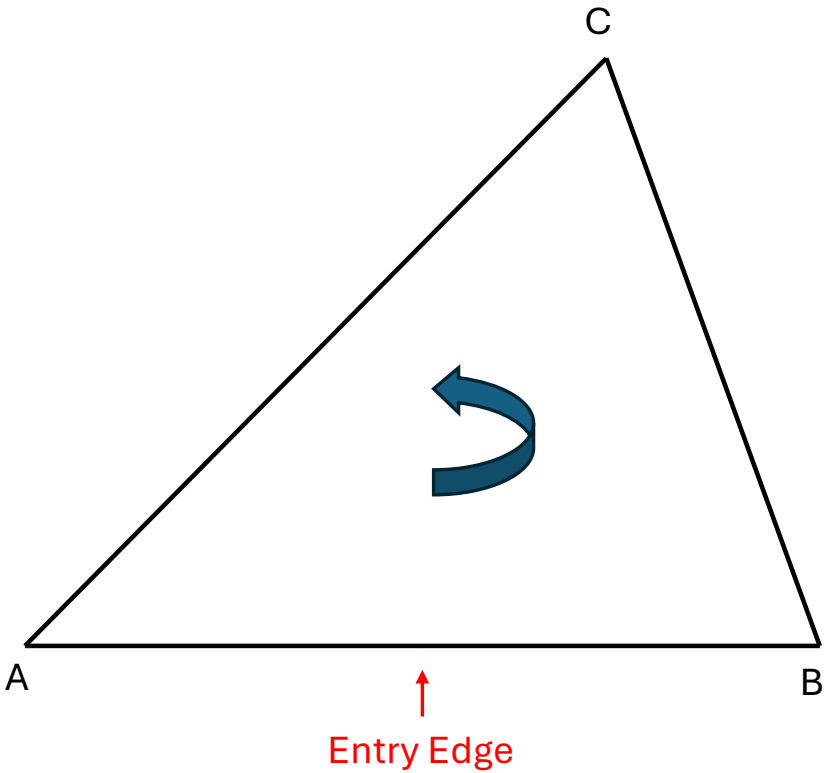


3D Triangle Mesh (2)

- How to “navigate” the mesh:

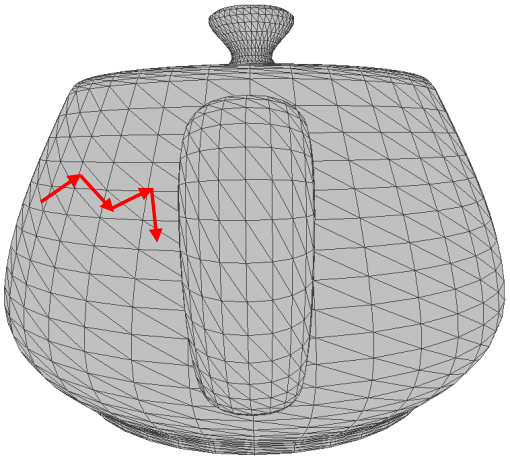


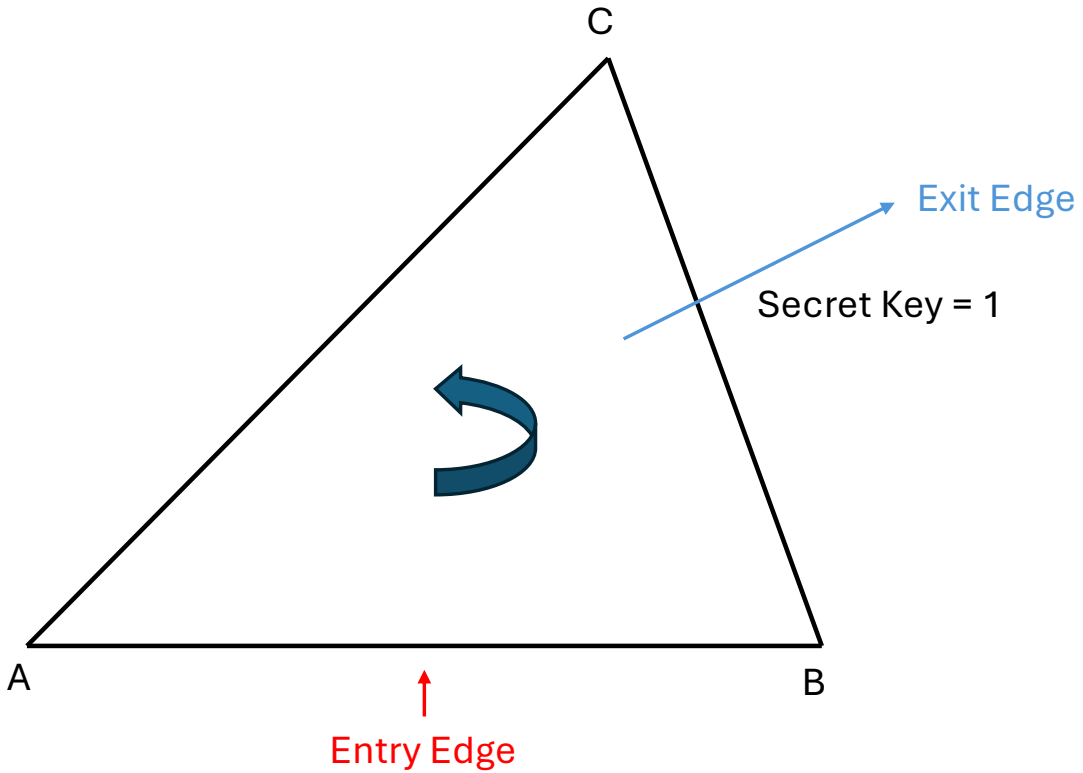
The starting triangle is selected according to geometric constraints



3D Triangle Mesh (2)

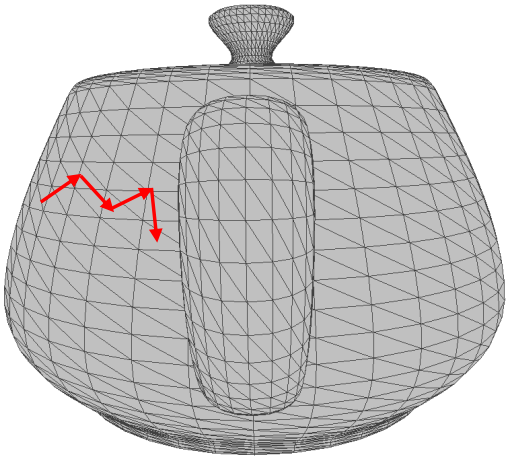
- How to “navigate” the mesh:

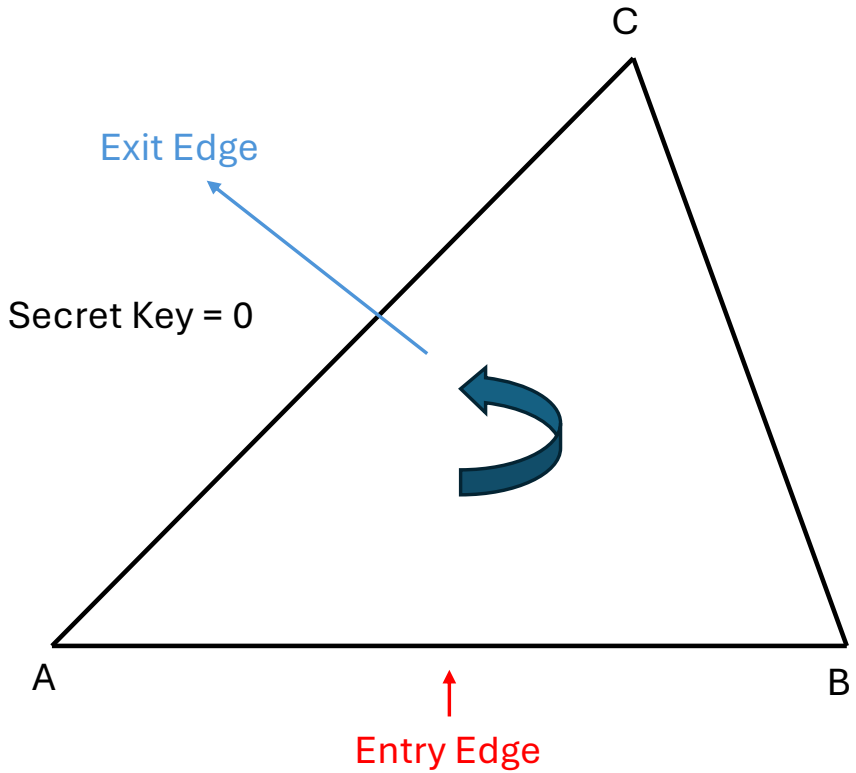




3D Triangle Mesh (2)

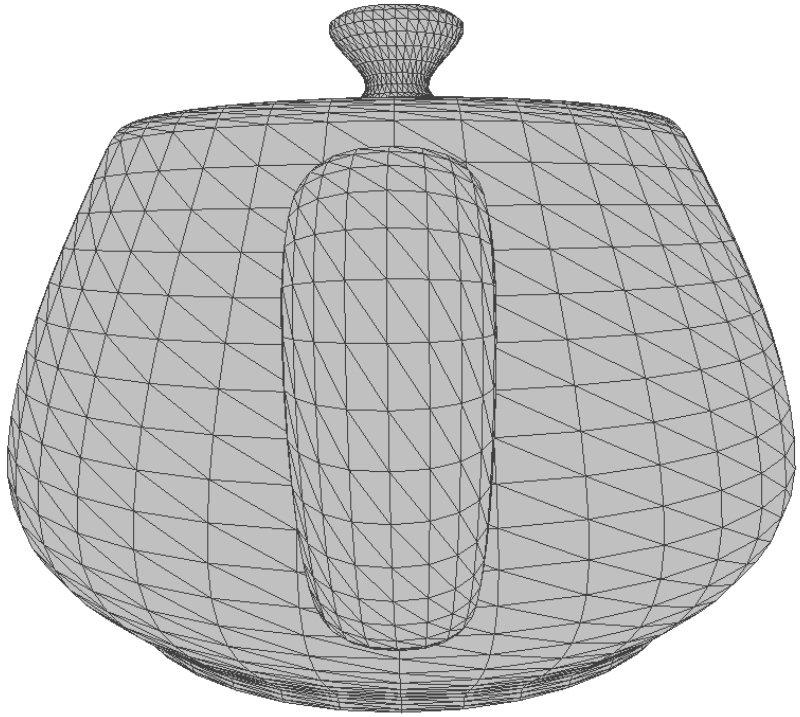
- How to “navigate” the mesh:





3D Triangle Mesh (3)

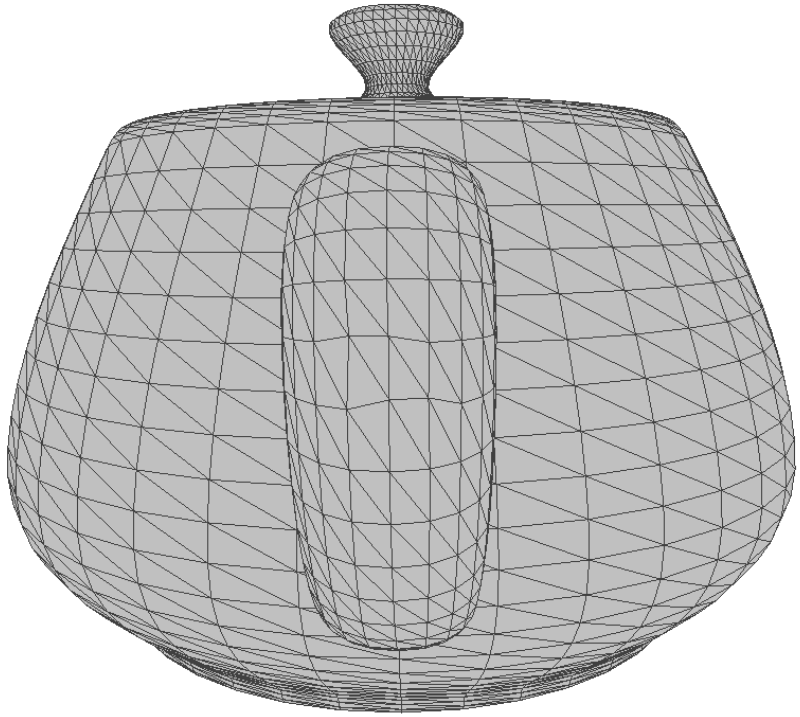
- Demo:



Examples/3D Triangle Mesh

3D Triangle Mesh (3)

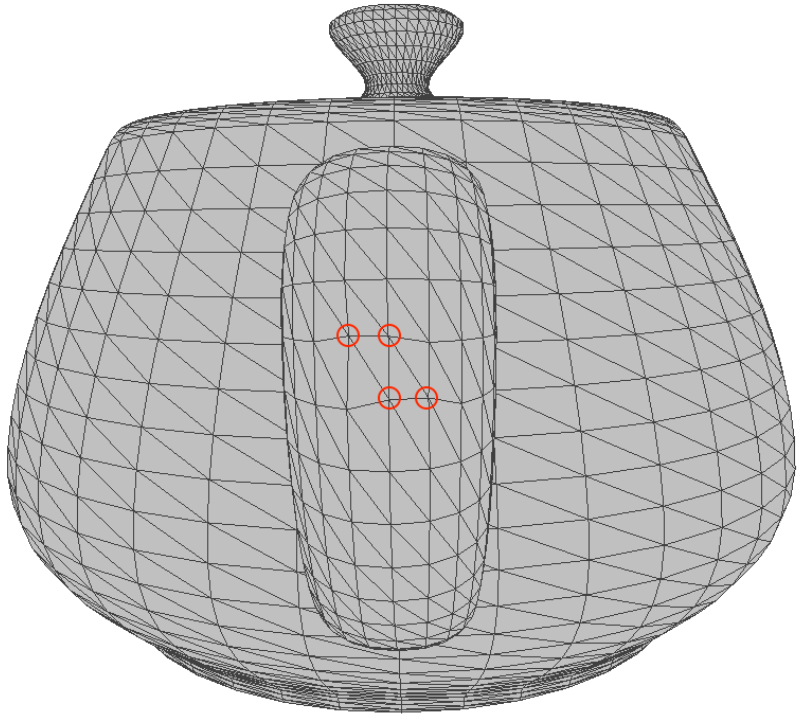
- Demo:



Examples/3D Triangle Mesh

3D Triangle Mesh (3)

- Demo:



Examples/3D Triangle Mesh

Network Traffic (1)

- Network traffic is **ubiquitous** as the Internet:
  - “glue” for microservices
  - contains sensitive data (even if encrypted)
  - re-routed by threat actors
  - processed by many defensive tools (even AI based)
- There is the need of techniques for:
  - traffic engineering
  - network design / troubleshoot
  - tracing malware, attackers, and botnets
  - identify endpoints even when encryption is used

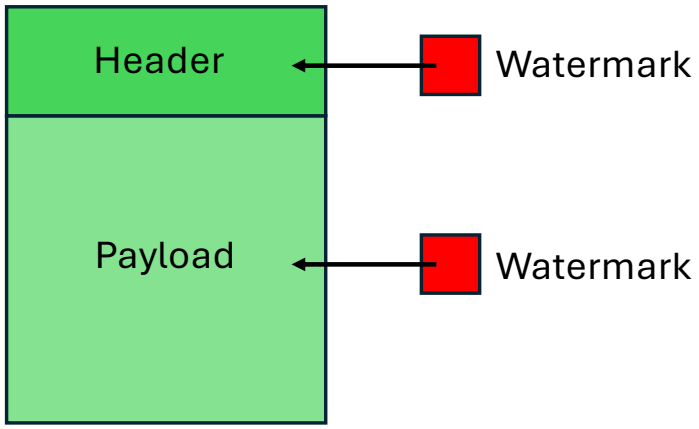
Network Traffic (2)

- Compared to other domains, **traffic watermarking** is quite **new**
- A good number of mechanisms is becoming available(*):
  - techniques are often borrowed from the literature on **covert channels**
- They are usually classified according to **where** the watermark is hidden

(*) A. Iacovazzi, Y. Elovici, “Network Flow Watermarking: A Survey”, IEEE Communications Surveys & Tutorials, Vol. 19, No. 1, pp. 512-530, First Quarter 2017

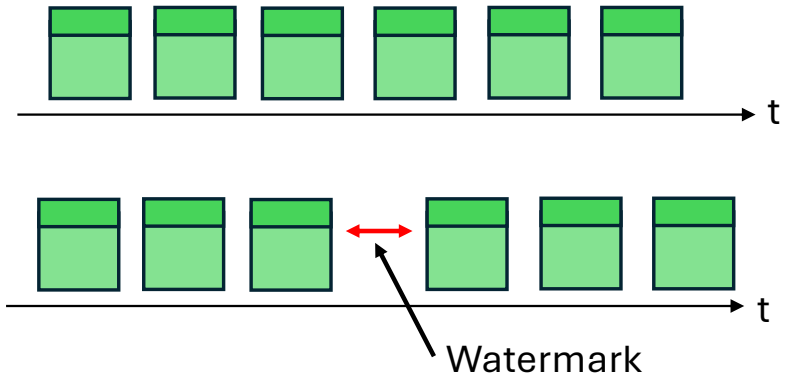
Network Traffic (2)

- Compared to other domains, **traffic watermarking** is quite **new**
- A good number of mechanisms is becoming available:
  - techniques are often borrowed from the literature on **covert channels**
- They are usually classified according to **where** the watermark is hidden
- **Content** based:
  - header
  - payload



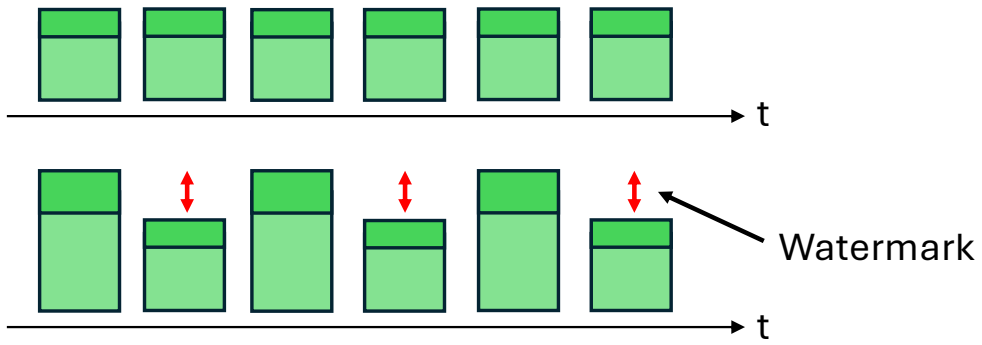
Network Traffic (2)

- Compared to other domains, **traffic watermarking** is quite **new**
- A good number of mechanisms is becoming available:
  - techniques are often borrowed from the literature on **covert channels**
- They are usually classified according to **where** the watermark is hidden
- **Timing** based:
  - delay
  - counting
  - statistical



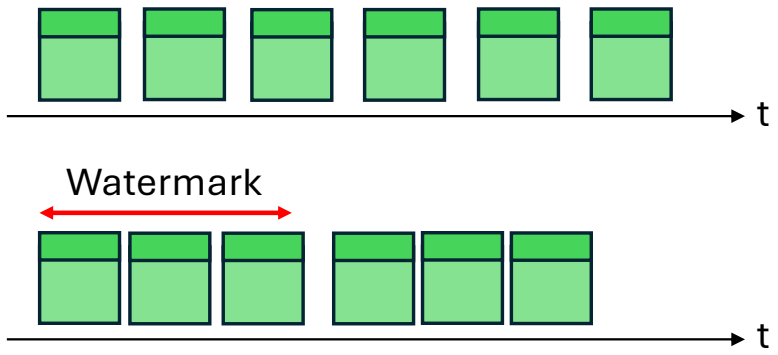
Network Traffic (2)

- Compared to other domains, **traffic watermarking** is quite **new**
- A good number of mechanisms is becoming available:
  - techniques are often borrowed from the literature on **covert channels**
- They are usually classified according to **where** the watermark is hidden
- **Size** based:
  - packet length
  - optional fields



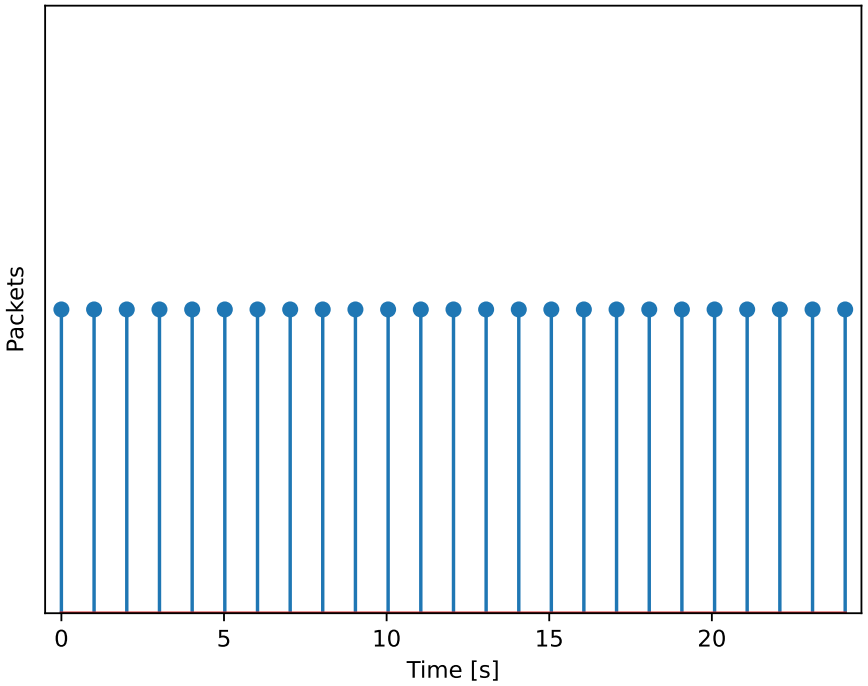
Network Traffic (2)

- Compared to other domains, **traffic watermarking** is quite **new**
- A good number of mechanisms is becoming available:
  - techniques are often borrowed from the literature on **covert channels**
- They are usually classified according to **where** the watermark is hidden
- **Rate** based:
  - throughput
  - packet rate

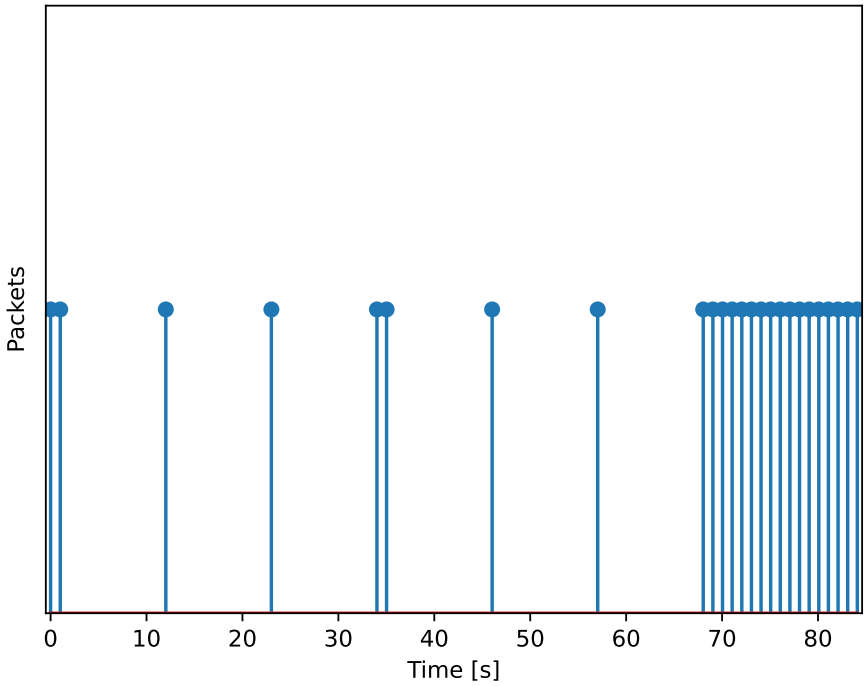


Network Traffic (3)

- Demo (timing watermark):



Inter-Packet Time of the Original Traffic

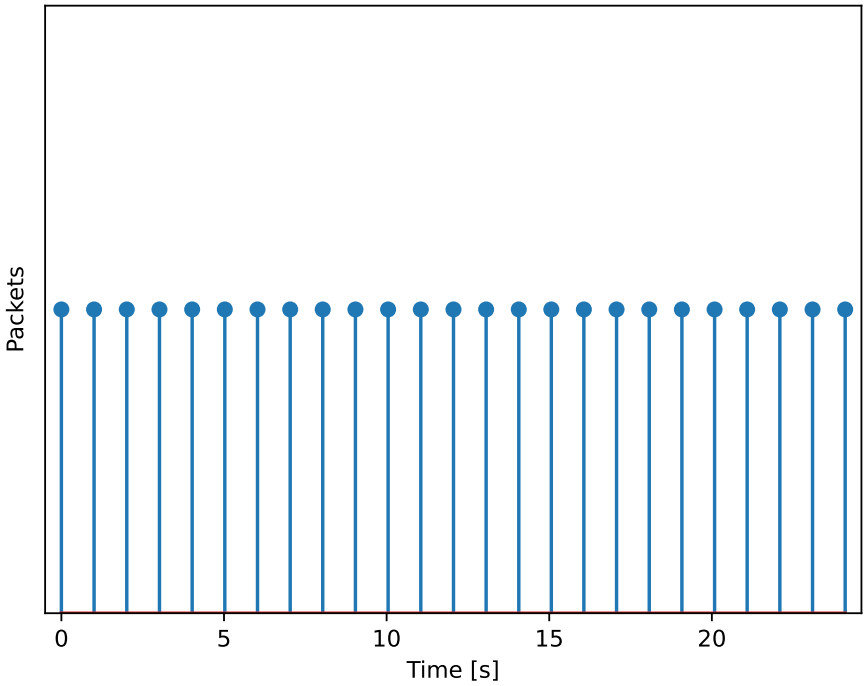


Inter-Packet Time of the Watermarked Traffic

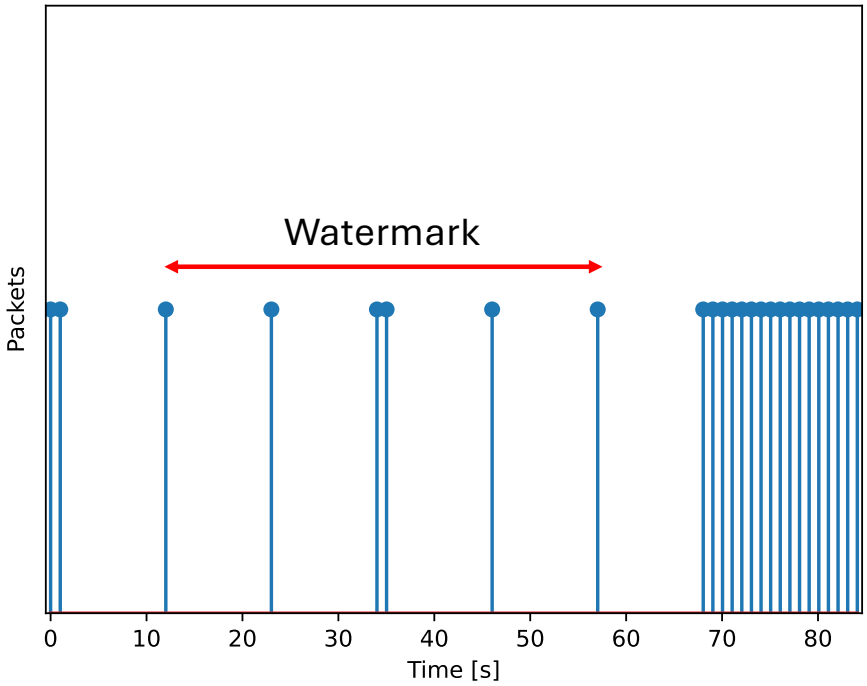
Examples/Network Traffic/Timing Method

Network Traffic (3)

- Demo (timing watermark):



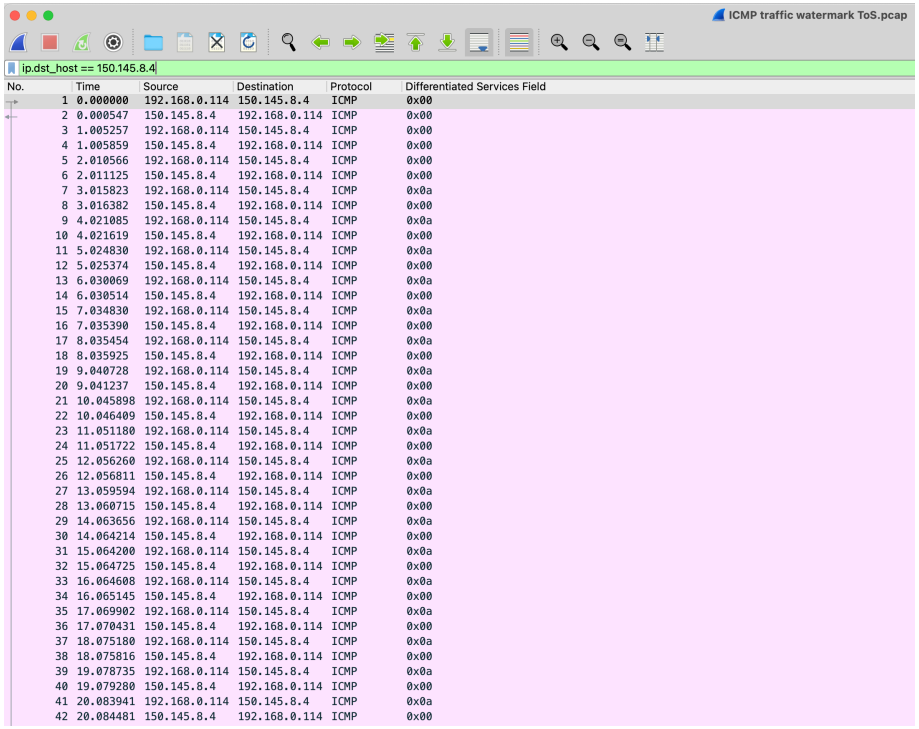
Inter-Packet Time of the Original Traffic



Inter-Packet Time of the Watermarked Traffic

Network Traffic (3)

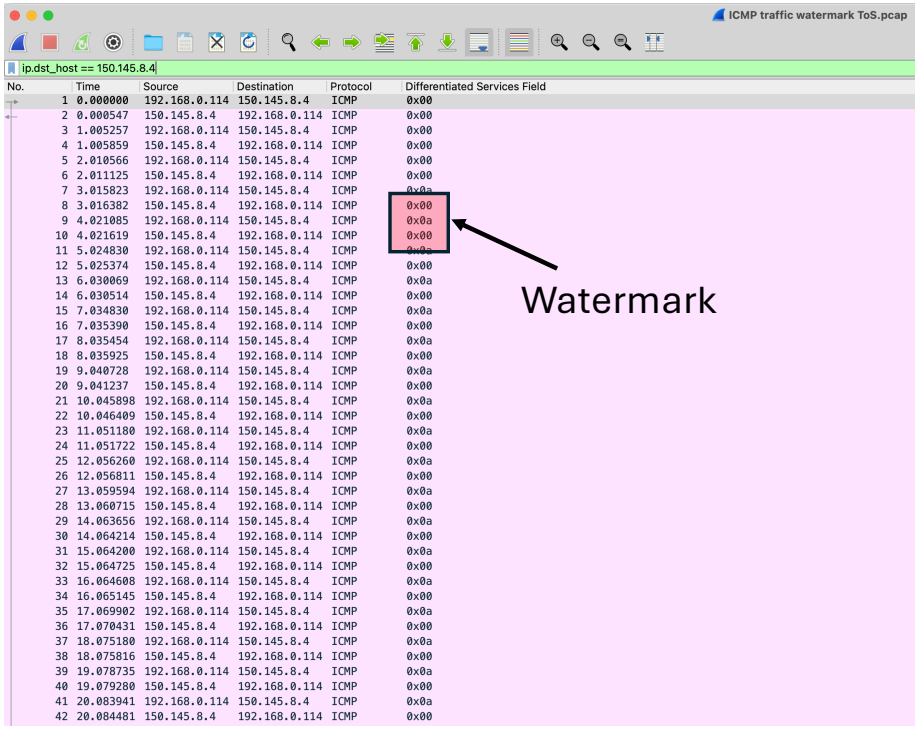
- Demo (content watermark):



No.	Time	Source	Destination	Protocol	Differentiated Services Field
1	0.000000	192.168.0.114	150.145.8.4	ICMP	0x00
2	0.000547	150.145.8.4	192.168.0.114	ICMP	0x00
3	1.005257	192.168.0.114	150.145.8.4	ICMP	0x00
4	1.005859	150.145.8.4	192.168.0.114	ICMP	0x00
5	2.010566	192.168.0.114	150.145.8.4	ICMP	0x00
6	2.011125	150.145.8.4	192.168.0.114	ICMP	0x00
7	3.015823	192.168.0.114	150.145.8.4	ICMP	0x0a
8	3.016382	150.145.8.4	192.168.0.114	ICMP	0x00
9	4.021085	192.168.0.114	150.145.8.4	ICMP	0x0a
10	4.021619	150.145.8.4	192.168.0.114	ICMP	0x00
11	5.024830	192.168.0.114	150.145.8.4	ICMP	0x0a
12	5.025374	150.145.8.4	192.168.0.114	ICMP	0x00
13	6.030069	192.168.0.114	150.145.8.4	ICMP	0x0a
14	6.030514	150.145.8.4	192.168.0.114	ICMP	0x00
15	7.034830	192.168.0.114	150.145.8.4	ICMP	0x0a
16	7.035390	150.145.8.4	192.168.0.114	ICMP	0x00
17	8.035454	192.168.0.114	150.145.8.4	ICMP	0x0a
18	8.035925	150.145.8.4	192.168.0.114	ICMP	0x00
19	9.040728	192.168.0.114	150.145.8.4	ICMP	0x0a
20	9.041237	150.145.8.4	192.168.0.114	ICMP	0x00
21	10.045898	192.168.0.114	150.145.8.4	ICMP	0x0a
22	10.046409	150.145.8.4	192.168.0.114	ICMP	0x00
23	11.051180	192.168.0.114	150.145.8.4	ICMP	0x0a
24	11.051722	150.145.8.4	192.168.0.114	ICMP	0x00
25	12.056260	192.168.0.114	150.145.8.4	ICMP	0x0a
26	12.056811	150.145.8.4	192.168.0.114	ICMP	0x00
27	13.059594	192.168.0.114	150.145.8.4	ICMP	0x0a
28	13.060715	150.145.8.4	192.168.0.114	ICMP	0x00
29	14.063656	192.168.0.114	150.145.8.4	ICMP	0x0a
30	14.064214	150.145.8.4	192.168.0.114	ICMP	0x00
31	15.064200	192.168.0.114	150.145.8.4	ICMP	0x0a
32	15.064725	150.145.8.4	192.168.0.114	ICMP	0x00
33	16.064608	192.168.0.114	150.145.8.4	ICMP	0x0a
34	16.065145	150.145.8.4	192.168.0.114	ICMP	0x00
35	17.069902	192.168.0.114	150.145.8.4	ICMP	0x0a
36	17.070431	150.145.8.4	192.168.0.114	ICMP	0x00
37	18.075180	192.168.0.114	150.145.8.4	ICMP	0x0a
38	18.075816	150.145.8.4	192.168.0.114	ICMP	0x00
39	19.078735	192.168.0.114	150.145.8.4	ICMP	0x0a
40	19.079280	150.145.8.4	192.168.0.114	ICMP	0x00
41	20.083941	192.168.0.114	150.145.8.4	ICMP	0x0a
42	20.084481	150.145.8.4	192.168.0.114	ICMP	0x00

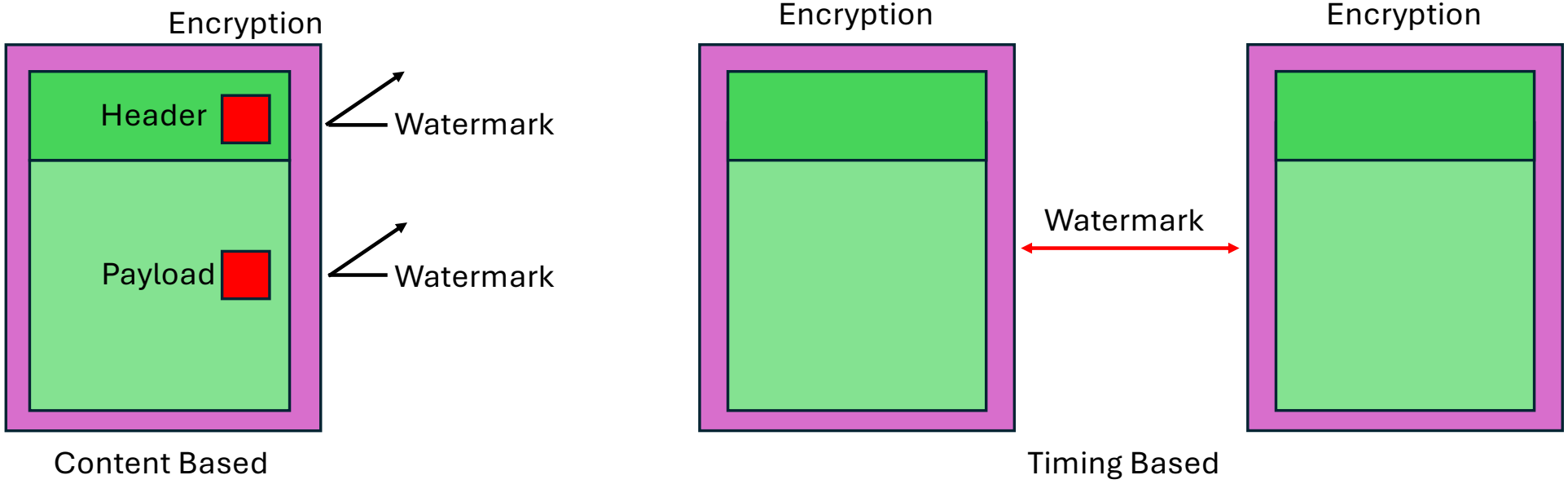
Network Traffic (3)

- Demo (content watermark):



Network Traffic (4)

- Each method has:
  - **pros** and **cons**
  - suitability for a specific **goal**
- Example:



Software (1)

- **Software** is today a **critical** asset (obviously):
  - piracy
  - reverse engineering
  - IP infringements
- With the advent of **AI**-based **code generators** or **assistants**:
  - new problems due to the ingestion of source code!
  - ability of generating malicious/unaudited code is magnified!
- Two main mechanisms for marking the software:
  - **static**: the watermark is hidden in code or assets (e.g., icons)
  - **dynamic**: the watermark is stored in the execution state of a program

Software (2)

- **Three** main exemplary sets of techniques
- **Easter Egg**:
  - simple and “catchy”
  - easy to locate
  - example: *about:mozilla*

Dan Forden

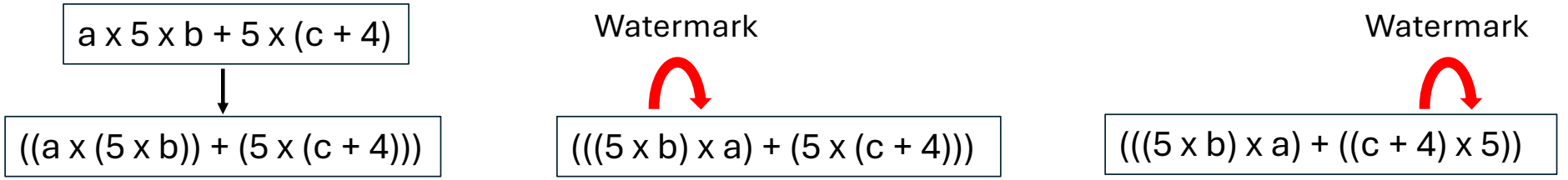
Software (2)

- **Three** main exemplary sets of techniques
- Easter Egg:
  - simple and “catchy”
  - easy to locate
  - example: *about:mozilla*
- **Code substitution** (or replacement):
  - general idea: substitute a portion of the code with a watermarked value
  - simple example: re-ordering of mathematical operations(*)

(*) J. Hamilton, S. Danicic, “A Survey of Static Software Watermarking”, World Congress on Internet Security, London, UK, pp. 100-107, Feb. 2011

Software (2)

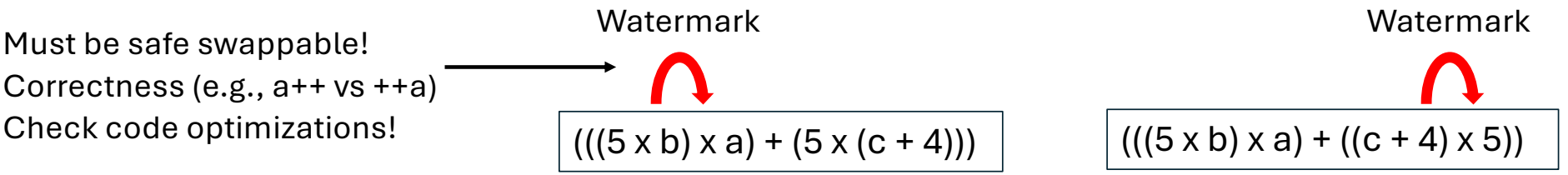
- **Three** main exemplary sets of techniques
- Easter Egg:
  - simple and “catchy”
  - easy to locate
  - example: *about:mozilla*
- **Code substitution** (or replacement):
  - general idea: substitute a portion of the code with a watermarked value
  - simple example: re-ordering of mathematical operations(*)



(*) J. Hamilton, S. Danicic, “A Survey of Static Software Watermarking”, World Congress on Internet Security, London, UK, pp. 100-107, Feb. 2011

Software (2)

- **Three** main exemplary sets of techniques
- Easter Egg:
  - simple and “catchy”
  - easy to locate
  - example: *about:mozilla*
- **Code substitution** (or replacement):
  - general idea: substitute a portion of the code with a watermarked value
  - simple example: re-ordering of mathematical operations(*)



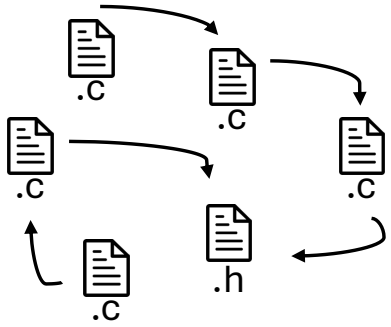
(*) J. Hamilton, S. Danicic, “A Survey of Static Software Watermarking”, World Congress on Internet Security, London, UK, pp. 100-107, Feb. 2011

Software (3)

- **Three** main exemplary sets of techniques
- Easter Egg:
  - simple and “catchy”
  - easy to locate
  - example: *about:mozilla*
- Code substitution (or replacement):
  - general idea: substitute a portion of the code with a watermarked value
  - simple example: re-ordering of mathematical operations(*)
- **Opaque predicates**:
  - borrowed from software protections and obfuscation

Opaque Predicates (1)

- Key idea:
  - create an expression that evaluates to either “true” or “false”
  - the outcome is **known** a-priori by the **programmer**
  - needs to be evaluated at run time
  - **difficult** to **understand** the intent of the code



Opaque Predicates (1)

- Key idea:
  - create an expression that evaluates to either “true” or “false”
  - the outcome is **known** a-priori by the **programmer**
  - needs to be evaluated at run time
  - **difficult** to **understand** the intent of the code

Opaque predicate
(d is always 8)!

Watermark

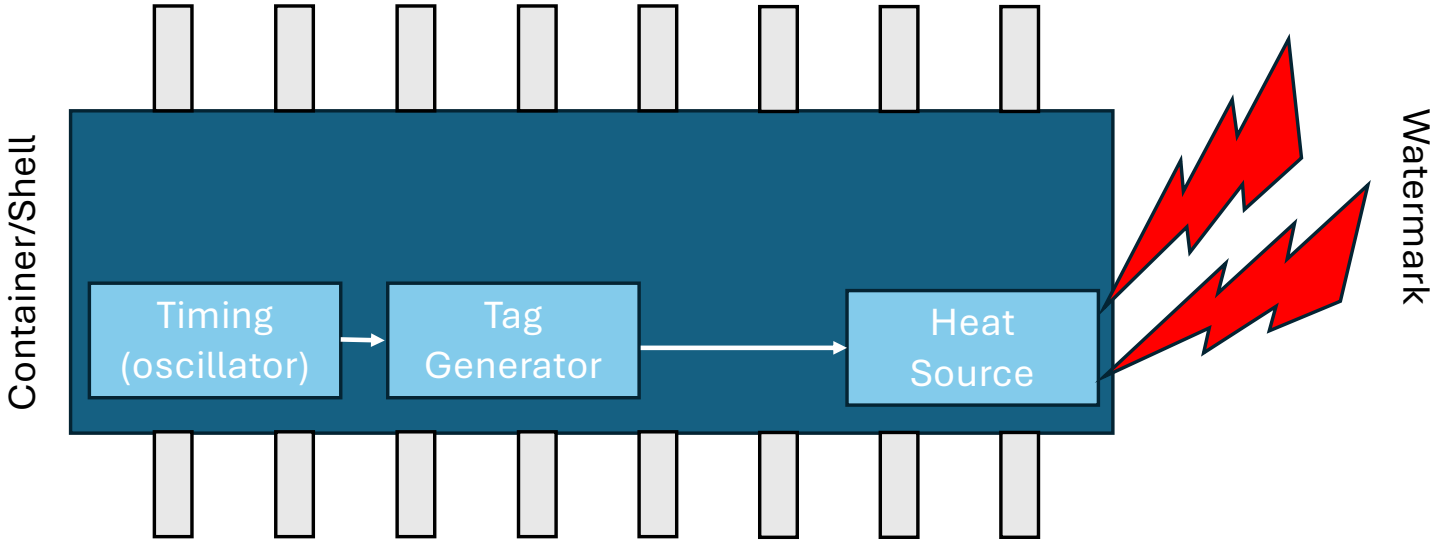
Examples/Opaque Predicate

Hardware (1)

- Custom hardware and “silicon” are now at the basis of:
  - IoT nodes
  - mobile/personal devices
  - CPU/GPU
  - AI-oriented architectures
- Progressive **co-design** of hardware and software for:
  - performances
  - sovereignty
  - greening
- **Good to hear it once** :)

Hardware (2)

- Only mention here a **simple** example:
  - a good starting point is (*) and the references therein



(*) C.-H. Chang, M. Potkonjak, L. Zhang, “Hardware IP Watermarking and Fingerprinting”, Secure System Design and Trustable Computing, pp. 329-368, Springer, 2016

AI Models (1)

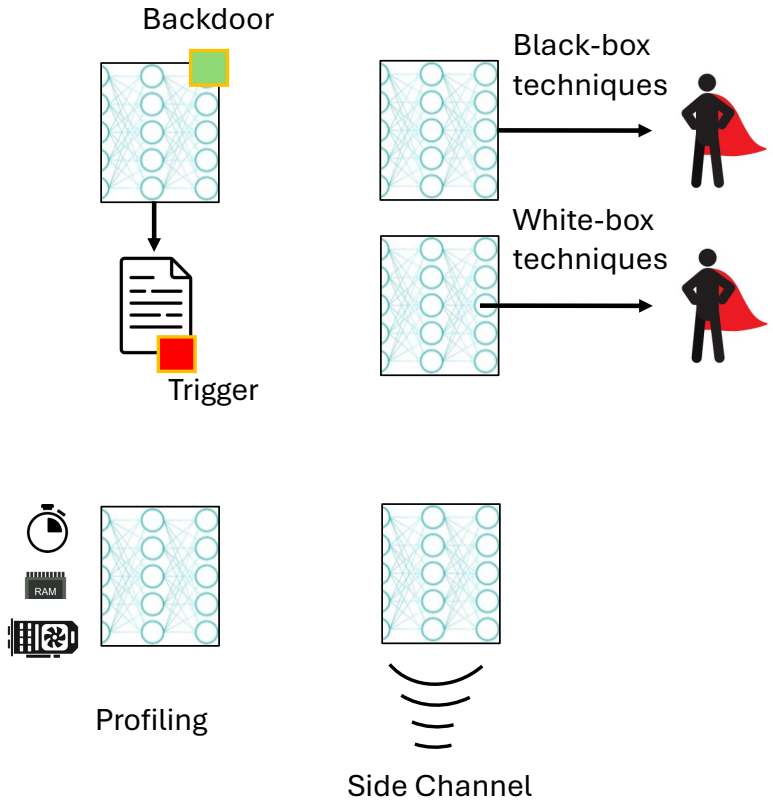
- Some **facts** on the machine learning **industry** in 2021:
  - it was valued $15.5B
  - the Netflix recommendation engine saves Netflix $1 billion per year
- **Watermarking** AI models is an **urgent need**:
  - …try to do a Google Search
  - many recent works still on Arxiv!
  - funded project

WHAM! - **W**atermarking **H**azards and novel perspectives in **A**dversarial **M**achine learning (PRIN - B53D23013340006)

Source: https://community.sap.com/t5/artificial-intelligence-and-machine-learning-blogs/protect-your-machine-learning-models-with-watermarking/ba-p/13519436

AI Models (2)

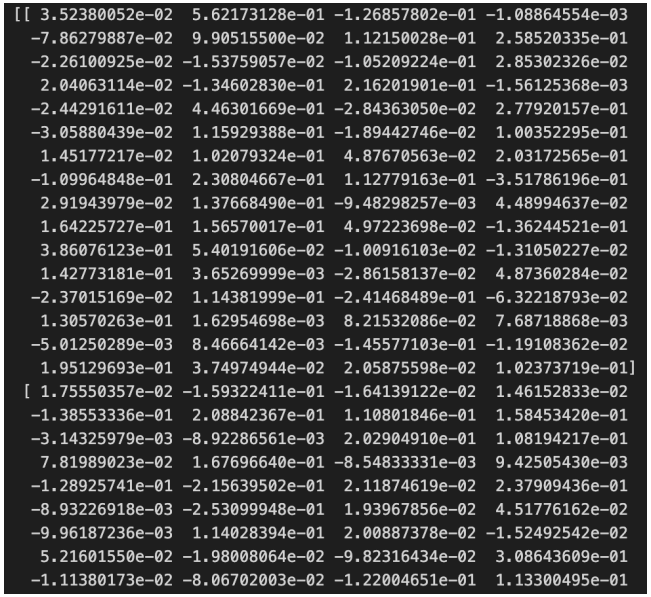
- Various techniques:
  - backdoors
  - profiling
  - side channels
  - white/black-box
  - …
- Example of a **white-box** technique:
  - it uses LSB (again: it is ubiquitous…)
  - it directly embeds data in weights
  - a bit naïve, just an example!
  - a more sophisticated approach is (*)



(*) Y. Nagai, Y Uchida, S. Sakazawa, “Digital Watermarking for Deep Neural Networks”, International Journal of Multimedia Information Retrieval, Vol. 7, pp. 3–16, Feb. 2018

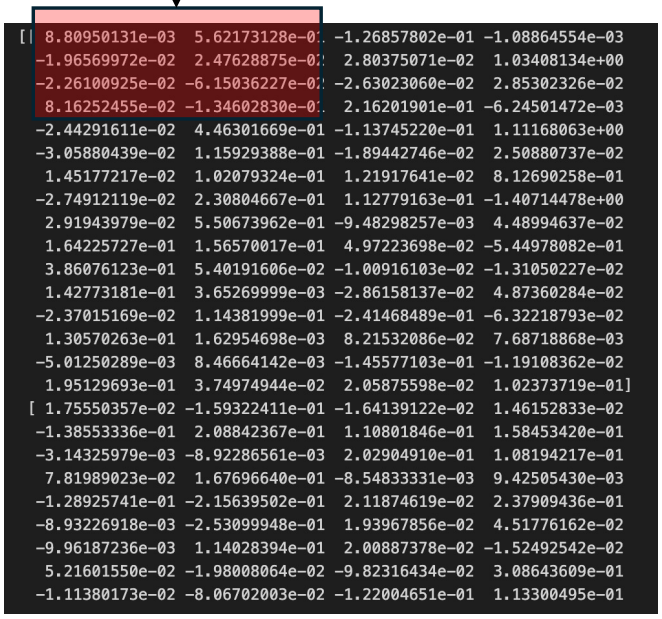
AI Models (2)

- Key idea:



Original Weights
(Trained Model)

Watermark



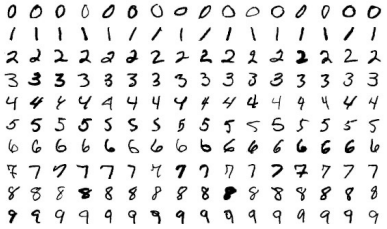
Watermarked Weights
(Trained Model)

AI Models (2)

- Key idea:

```
[[ 3.52380052e-02  5.62173128e-01 -1.26857802e-01 -1.08864554e-03
  -7.86279887e-02  9.90515500e-02  1.12150028e-01  2.58520335e-01
  -2.26100925e-02 -1.53759057e-02 -1.05209224e-01  2.85302326e-02
   2.04063114e-02 -1.34602830e-01  2.16201901e-01 -1.56125368e-03
  -2.44291611e-02  4.46301669e-01 -2.84363050e-02  2.77920157e-01
  -3.05880439e-02  1.15929388e-01 -1.89442746e-02  1.00352295e-01
   1.45177217e-02  1.02079324e-01  4.87670563e-02  2.03172565e-01
  -1.09964848e-01  2.30804667e-01  1.12779163e-01 -3.51786196e-01
   2.91943979e-02  1.37668490e-01 -9.48298257e-03  4.48994637e-02
   1.64225727e-01  1.56570017e-01  4.97223698e-02 -1.36244521e-01
   3.86076123e-01  5.40191606e-02 -1.00916103e-02 -1.31050227e-02
   1.42773181e-01  3.65269999e-03 -2.86158137e-02  4.87360284e-02
  -2.37015169e-02  1.14381999e-01 -2.41468489e-01 -6.32218793e-02
   1.30570263e-01  1.62954698e-03  8.21532086e-02  7.68718868e-03
  -5.01250289e-03  8.46664142e-03 -1.45577103e-01 -1.19108362e-02
   1.95129693e-01  3.74974944e-02  2.05875598e-02  1.02373719e-01]
 [ 1.75550357e-02 -1.59322411e-01 -1.64139122e-02  1.46152833e-02
  -1.38553336e-01  2.08842367e-01  1.10801846e-01  1.58453420e-01
  -3.14325979e-03 -8.92286561e-03  2.02904910e-01  1.08194217e-01
   7.81989023e-02  1.67696640e-01 -8.54833331e-03  9.42505430e-03
  -1.28925741e-01 -2.15639502e-01  2.11874619e-02  2.37909436e-01
  -8.93226918e-03 -2.53099948e-01  1.93967856e-02  4.51776162e-02
  -9.96187236e-03  1.14028394e-01  2.00887378e-02 -1.52492542e-02
   5.21601550e-02 -1.98008064e-02 -9.82316434e-02  3.08643609e-01
  -1.11380173e-02 -8.06702003e-02 -1.22004651e-01  1.13300495e-01
```

Original Weights
(Trained Model)



MNIST Dataset

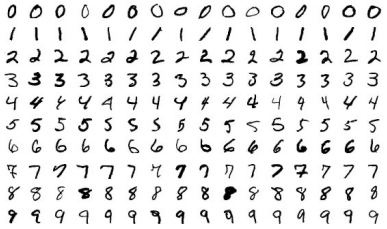
Accuracy: 98.58%

AI Models (2)

- Key idea:

```
[[ 8.80950131e-03  5.62173128e-01 -1.26857802e-01 -1.08864554e-03
  -1.96569972e-02  2.47628875e-02  2.80375071e-02  1.03408134e+00
  -2.26100925e-02 -6.15036227e-02 -2.63023060e-02  2.85302326e-02
   8.16252455e-02 -1.34602830e-01  2.16201901e-01 -6.24501472e-03
  -2.44291611e-02  4.46301669e-01 -1.13745220e-01  1.11168063e+00
  -3.05880439e-02  1.15929388e-01 -1.89442746e-02  2.50880737e-02
   1.45177217e-02  1.02079324e-01  1.21917641e-02  8.12690258e-01
  -2.74912119e-02  2.30804667e-01  1.12779163e-01 -1.40714478e+00
   2.91943979e-02  5.50673962e-01 -9.48298257e-03  4.48994637e-02
   1.64225727e-01  1.56570017e-01  4.97223698e-02 -5.44978082e-01
   3.86076123e-01  5.40191606e-02 -1.00916103e-02 -1.31050227e-02
   1.42773181e-01  3.65269999e-03 -2.86158137e-02  4.87360284e-02
  -2.37015169e-02  1.14381999e-01 -2.41468489e-01 -6.32218793e-02
   1.30570263e-01  1.62954698e-03  8.21532086e-02  7.68718868e-03
  -5.01250289e-03  8.46664142e-03 -1.45577103e-01 -1.19108362e-02
   1.95129693e-01  3.74974944e-02  2.05875598e-02  1.02373719e-01]
 [ 1.75550357e-02 -1.59322411e-01 -1.64139122e-02  1.46152833e-02
  -1.38553336e-01  2.08842367e-01  1.10801846e-01  1.58453420e-01
  -3.14325979e-03 -8.92286561e-03  2.02904910e-01  1.08194217e-01
   7.81989023e-02  1.67696640e-01 -8.54833331e-03  9.42505430e-03
  -1.28925741e-01 -2.15639502e-01  2.11874619e-02  2.37909436e-01
  -8.93226918e-03 -2.53099948e-01  1.93967856e-02  4.51776162e-02
  -9.96187236e-03  1.14028394e-01  2.00887378e-02 -1.52492542e-02
   5.21601550e-02 -1.98008064e-02 -9.82316434e-02  3.08643609e-01
  -1.11380173e-02 -8.06702003e-02 -1.22004651e-01  1.13300495e-01
```

Watermarked Weights
(Trained Model)



MNIST Dataset

Accuracy: 94.68%

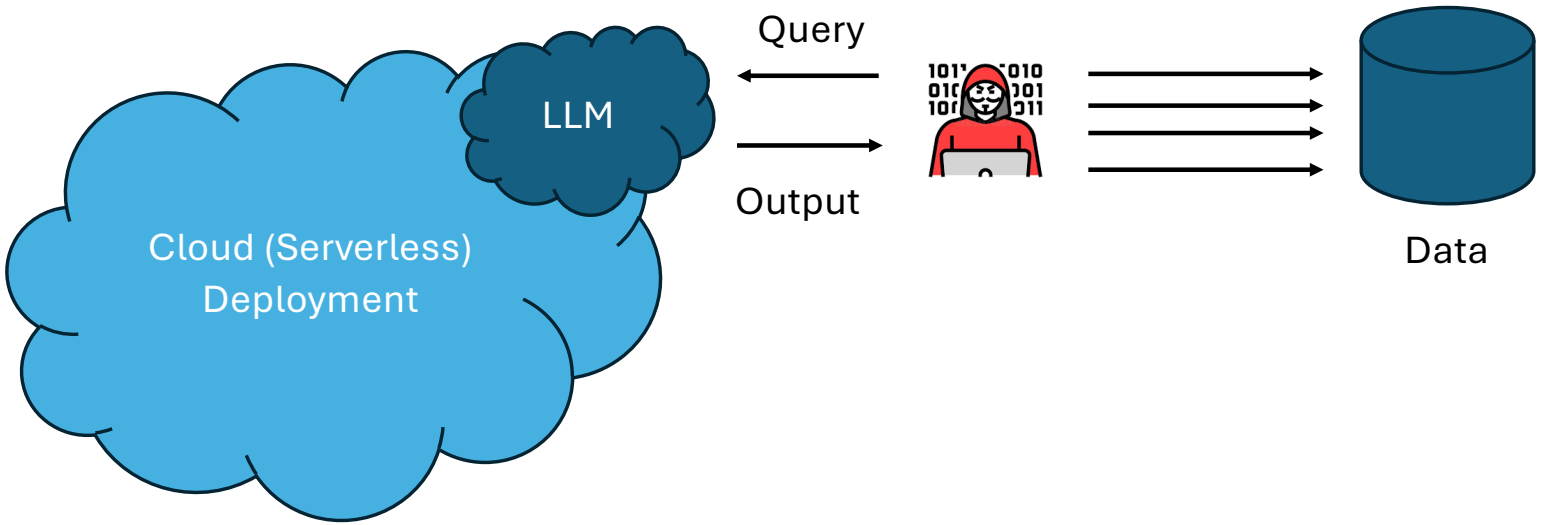
To reduce the impact of the watermark a possible approach is to use a “dropout” mechanism (before the FC2 layer) to induce an ensemble behavior.
The accuracy increases again to ~98%!

AI Outputs (1)

- Large Language Models (LLMs) spawned **new services**:
  - chatbots
  - financial analysis
  - content generation
  - code generation
  - …
- Commercial LLMs offers some form of APIs:
  - to **avoid training**, which is expensive and time-consuming
  - yet, an attacker can perform **multiple queries** to imitate the model
- We will threat LLM as a synecdoche of as-a-Service frameworks

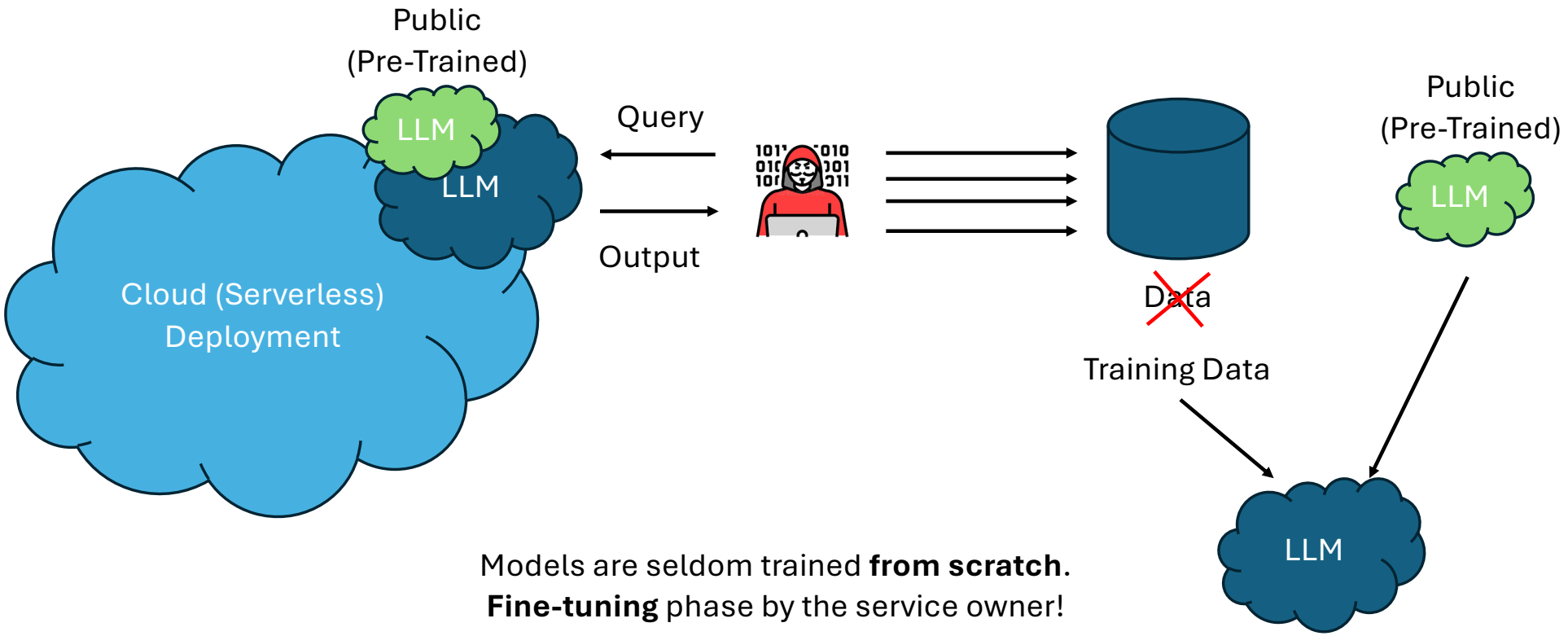
AI Outputs (2)

- General attack model:



AI Outputs (2)

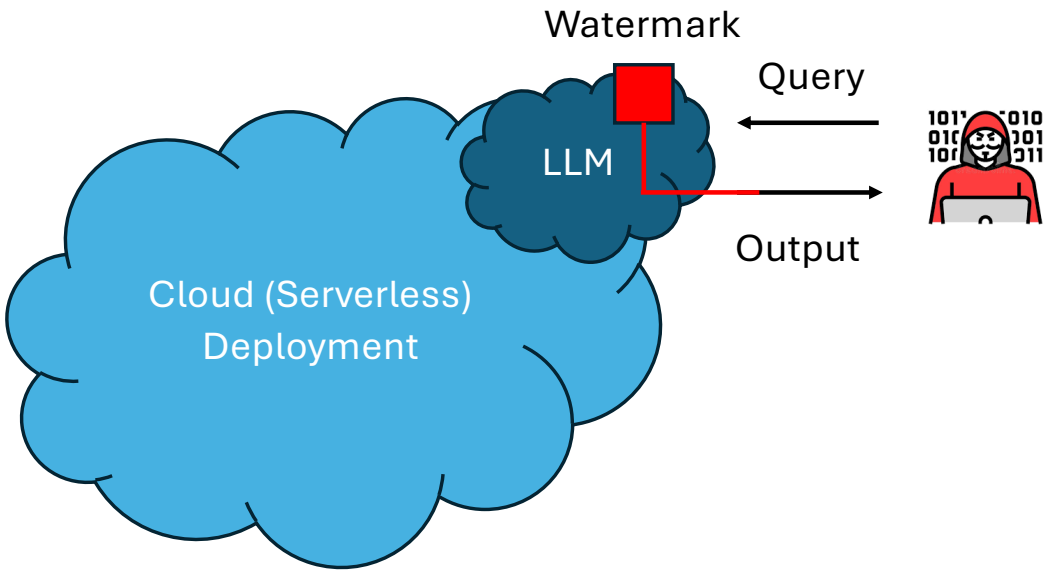
- General attack model:



Models are seldom trained **from scratch**.
Fine-tuning phase by the service owner!

AI Outputs (2)

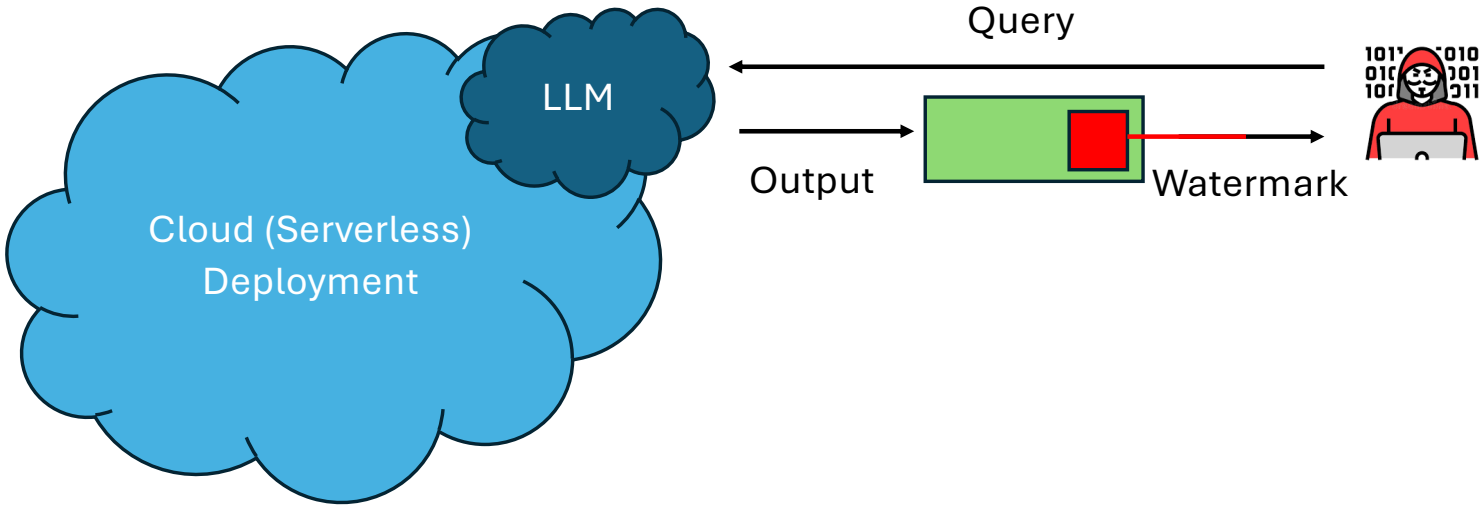
- General attack model:



"Soft" Watermark
(Requires to intercept and change the model to act in **white-box** manner)

AI Outputs (2)

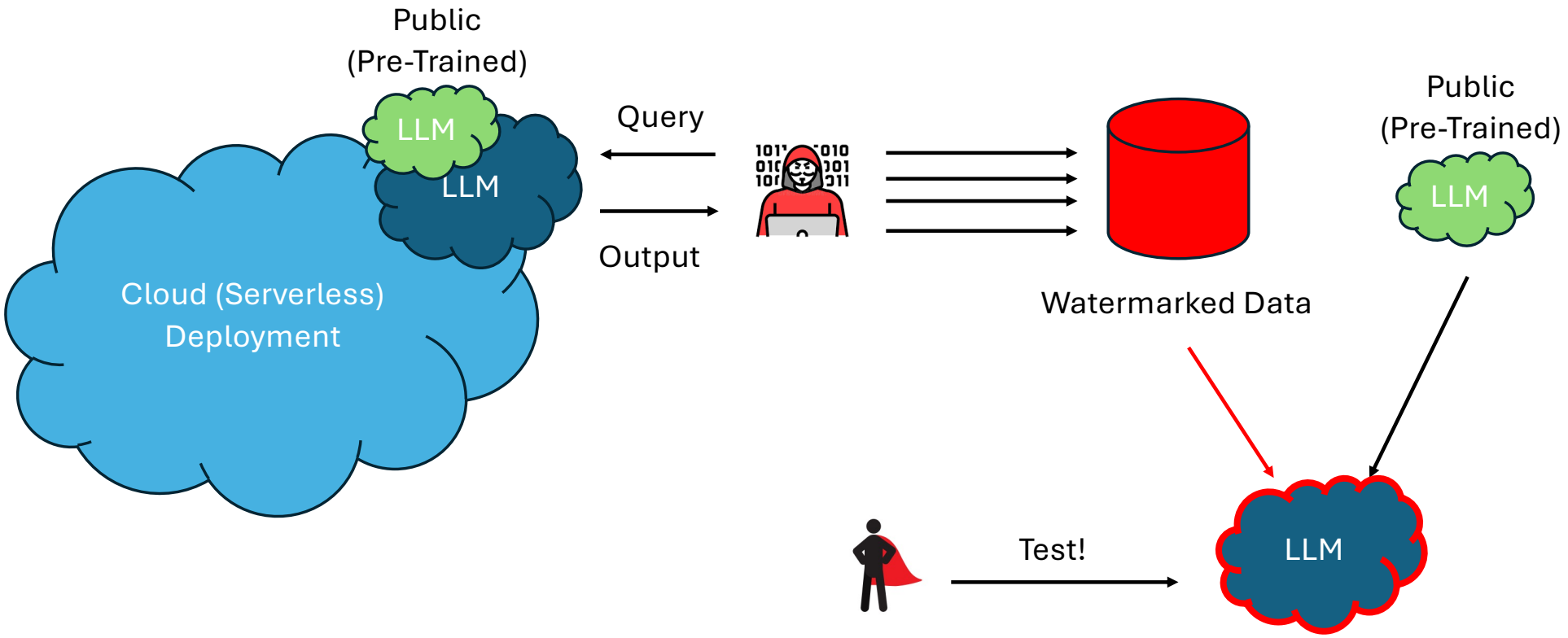
- General attack model:



"Hard" Watermark
(Considers the LLM/AI a **black-box**)

AI Outputs (2)

- General attack model:



AI Outputs (3)

- Example of a **black-box** technique:
  - for marking LLM-based code generation tools (very popular!)
  - from Li, Wang, Wang and Gao(*)

(*) Z. Li, C. Wang, S. Wang, C. Gao, “Protecting Intellectual Property of Large Language Model-Based Code Generation APIs via Watermarks”, Proc. of the ACM SIGSAC Conference on Computer and Communications Security, pp. 2336 – 2350, Nov. 2023

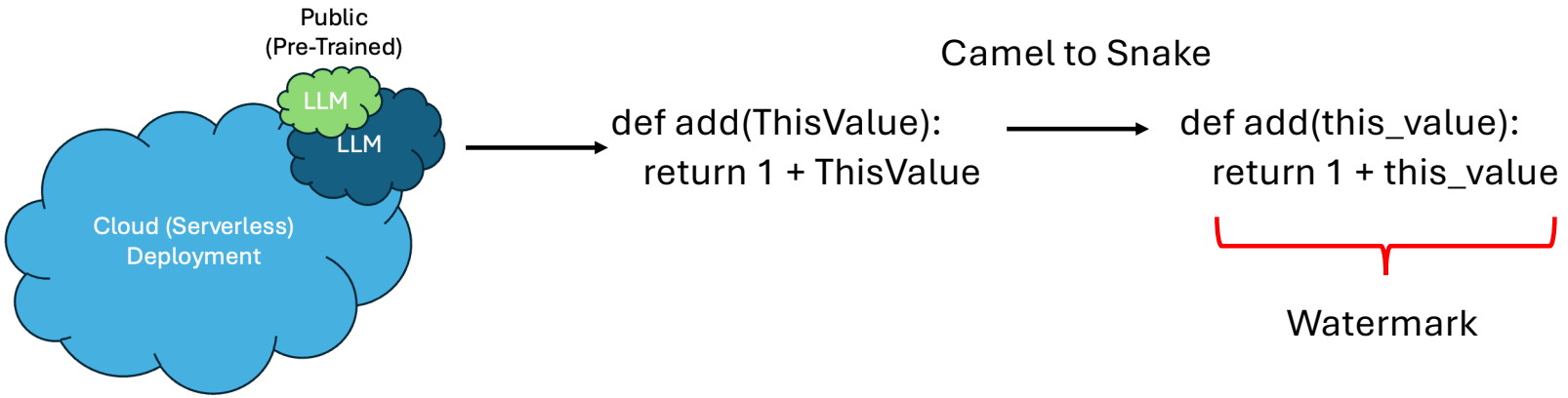
AI Outputs (3)

- Example of a **black-box** technique:
  - for marking LLM-based code generation tools (very popular!)
  - from Li, Wang, Wang and Gao(*)
- Key idea:
  - AI-generated code is changed by replacing tokens with synonyms
  - the process is iterated to match a target distribution (for test)
  - many modern programming languages can be altered without changing their semantics

(*) Z. Li, C. Wang, S. Wang, C. Gao, “Protecting Intellectual Property of Large Language Model-Based Code Generation APIs via Watermarks”, Proc. of the ACM SIGSAC Conference on Computer and Communications Security, pp. 2336 – 2350, Nov. 2023

AI Outputs (3)

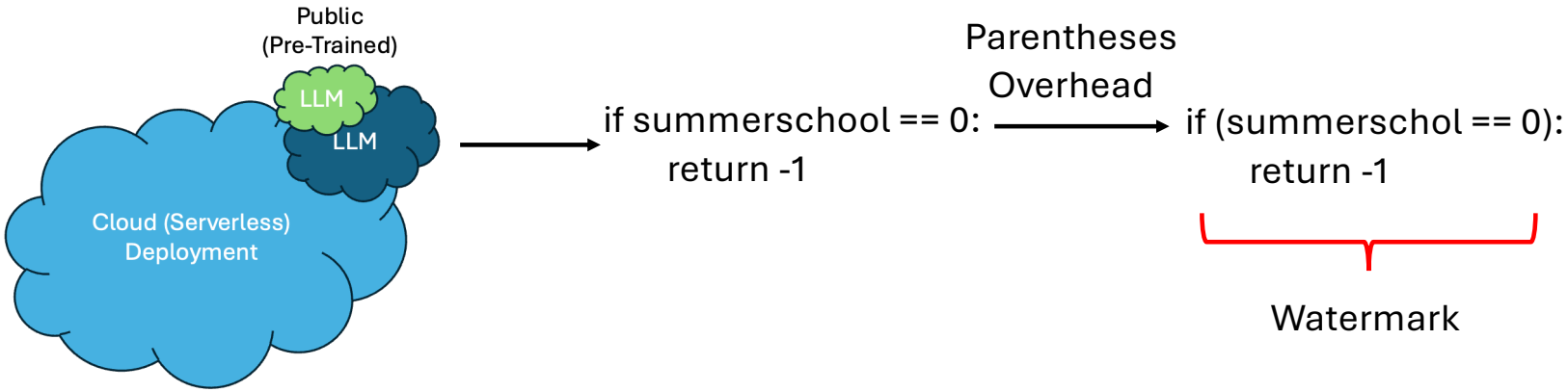
- Example of a **black-box** technique:
  - for marking LLM-based code generation tools (very popular!)
  - from Li, Wang, Wang and Gao(*)
- Key idea:



(*) Z. Li, C. Wang, S. Wang, C. Gao, "Protecting Intellectual Property of Large Language Model-Based Code Generation APIs via Watermarks", Proc. of the ACM SIGSAC Conference on Computer and Communications Security, pp. 2336 – 2350, Nov. 2023

AI Outputs (3)

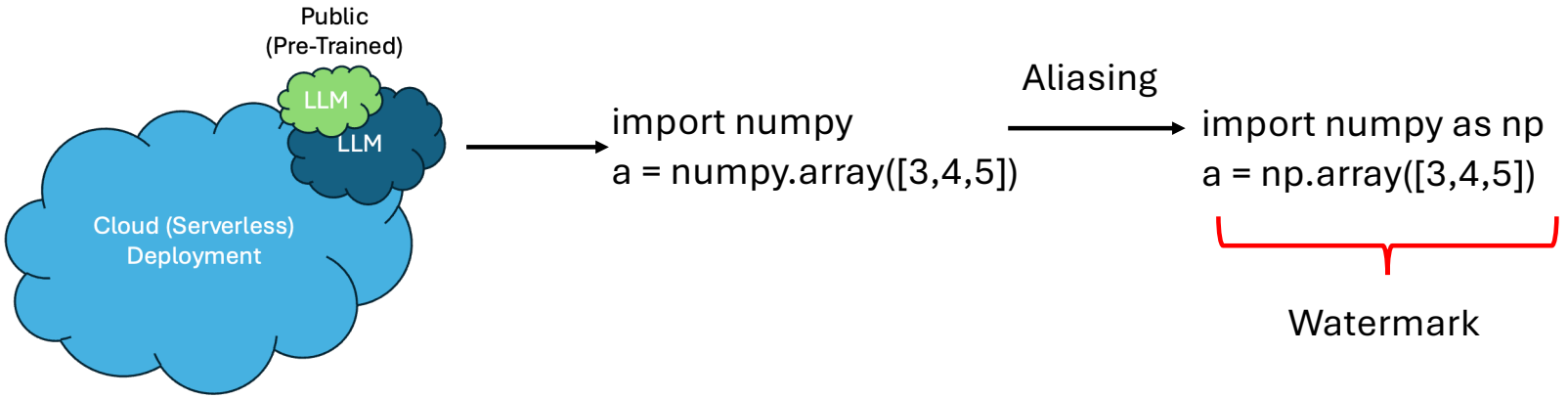
- Example of a **black-box** technique:
  - for marking LLM-based code generation tools (very popular!)
  - from Li, Wang, Wang and Gao(*)
- Key idea:



(*) Z. Li, C. Wang, S. Wang, C. Gao, "Protecting Intellectual Property of Large Language Model-Based Code Generation APIs via Watermarks", Proc. of the ACM SIGSAC Conference on Computer and Communications Security, pp. 2336 – 2350, Nov. 2023

AI Outputs (3)

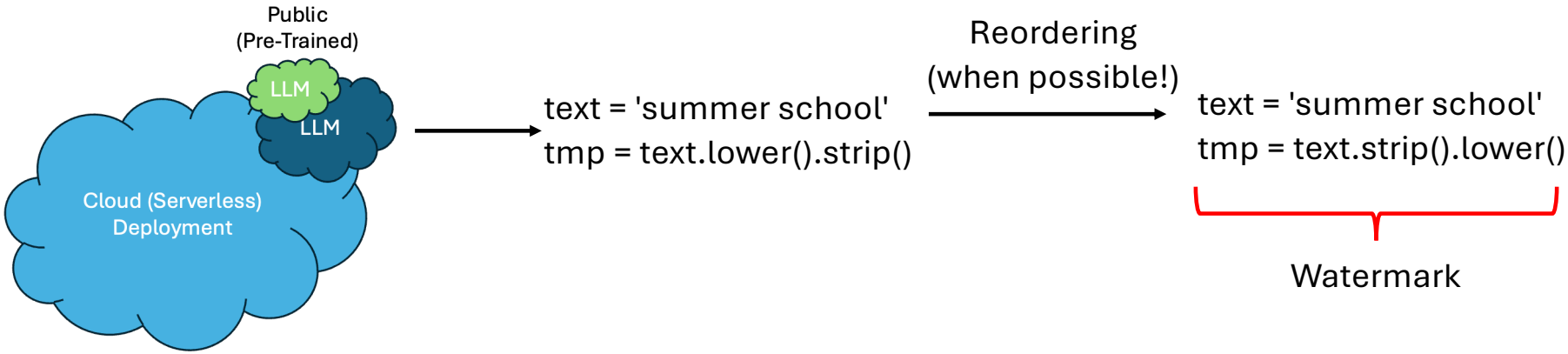
- Example of a **black-box** technique:
  - for marking LLM-based code generation tools (very popular!)
  - from Li, Wang, Wang and Gao(*)
- Key idea:



(*) Z. Li, C. Wang, S. Wang, C. Gao, "Protecting Intellectual Property of Large Language Model-Based Code Generation APIs via Watermarks", Proc. of the ACM SIGSAC Conference on Computer and Communications Security, pp. 2336 – 2350, Nov. 2023

AI Outputs (3)

- Example of a **black-box** technique:
  - for marking LLM-based code generation tools (very popular!)
  - from Li, Wang, Wang and Gao(*)
- Key idea:

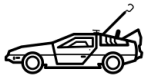


(*) Z. Li, C. Wang, S. Wang, C. Gao, "Protecting Intellectual Property of Large Language Model-Based Code Generation APIs via Watermarks", Proc. of the ACM SIGSAC Conference on Computer and Communications Security, pp. 2336 – 2350, Nov. 2023

(Some) Attacks

Attacks (1)

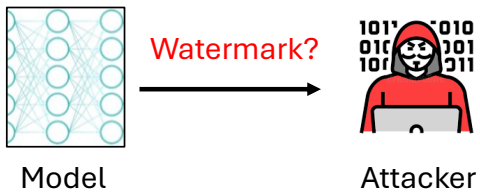
- Attacks heavily depend on:
  - the **type** of watermark
  - the protected media/content, i.e., the **carrier**
- Manipulations could be **intentional** or **unintentional**
- Several **attack models** are possible:
  - different degree of sophistication
  - exploiting diverse attack surfaces
  - tweaked for specific contexts, e.g., AI vs cloud
- Attacks can be “chained” to implement advanced **offensive templates**



Running out of time: attack models for many AI scenarios have been nicely described in: F. Boenisch, “A Systematic Review on Model Watermarking for Neural Networks”, Frontiers in Big Data, Vol. 4, pp. 4:1 – 4:16, Nov. 2021

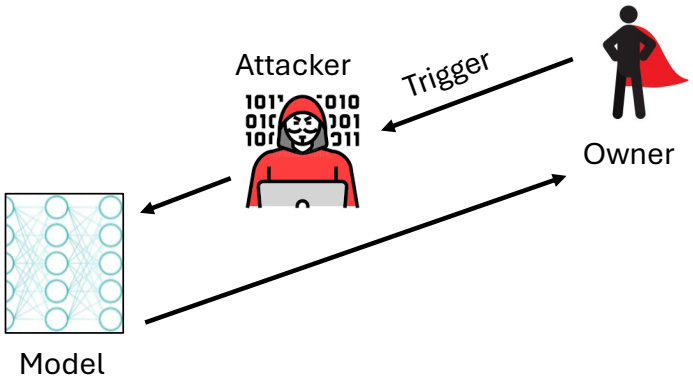
Attacks (2)

- The most important attack models are:
  - **detection**: attacker can discover the presence of a watermark

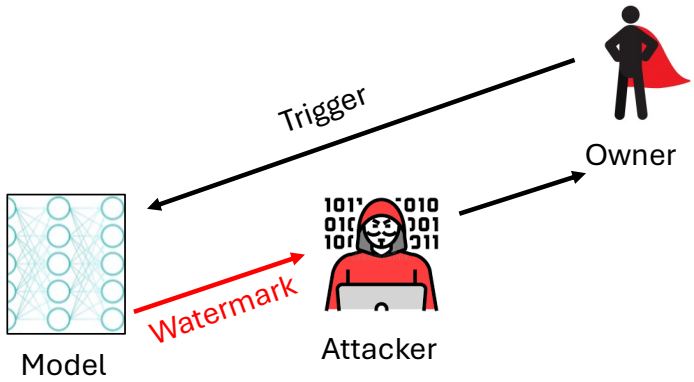


Attacks (2)

- The most important attack models are:
  - detection: attacker can discover the presence of a watermark
  - **suppression**: attacker can impede to access the watermark



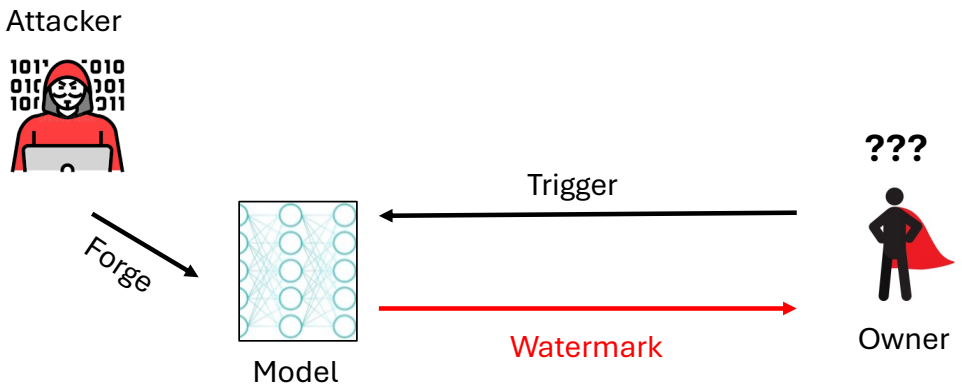
Trigger Suppression



Watermark Suppression

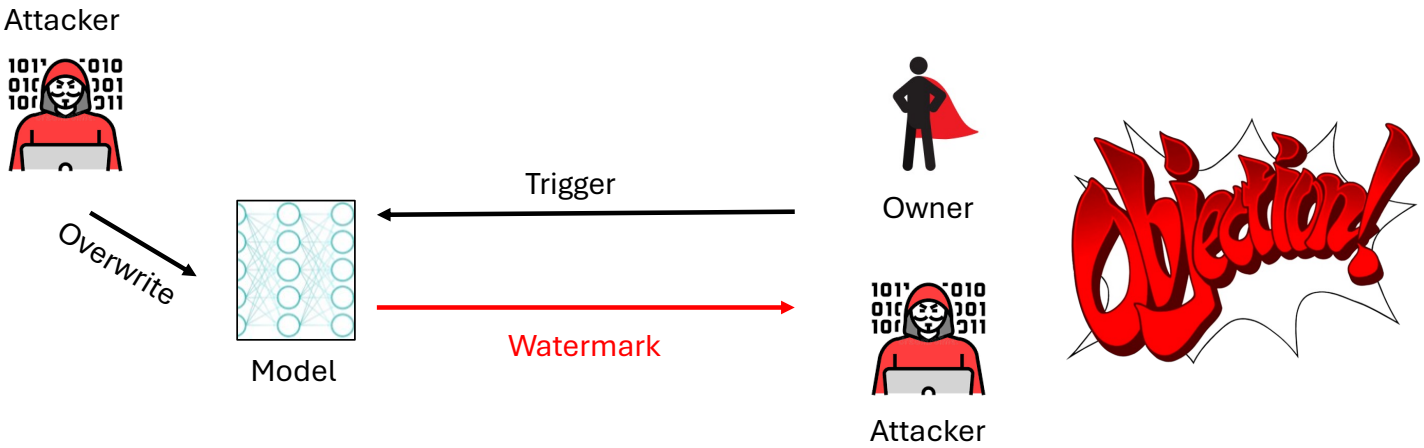
Attacks (2)

- The most important attack models are:
  - detection: attacker can discover the presence of a watermark
  - suppression: attacker can impede to access the watermark
  - **forging**: attacker can render the watermark ambiguous



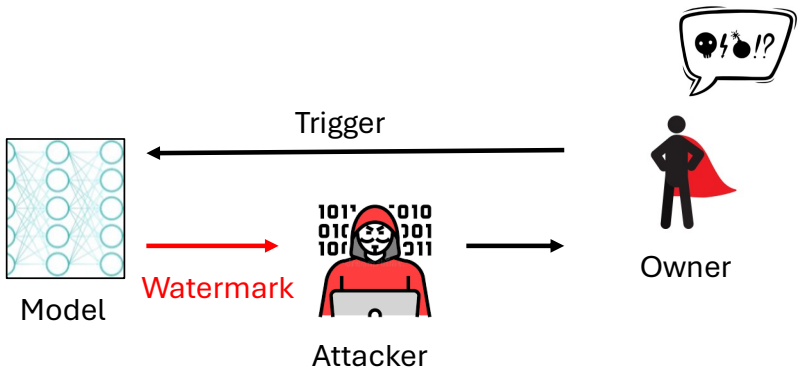
Attacks (2)

- The most important attack models are:
  - detection: attacker can discover the presence of a watermark
  - suppression: attacker can impede to access the watermark
  - forging: attacker can render the watermark ambiguous
  - **overwrite**: attacker can substitute the watermark to pretend ownership



Attacks (2)

- The most important attack models are:
  - detection: attacker can discover the presence of a watermark
  - suppression: attacker can impede to access the watermark
  - forging: attacker can render the watermark ambiguous
  - overwrite: attacker can substitute the watermark to pretend ownership
  - **removal**: attacker can prevent the use of watermark



Attacks (3)

- Removal attacks are the most popular, especially for:
  - multimedia
  - digital objects
  - 3D mesh
  - network traffic
- The main classes of **attacks** are:
  - additive noise
  - (pseudo) random alterations
  - remove frequencies via filtering
  - cropping, rotation, and scaling
  - smoothing and simplification
  - compression or transcoding
  - statistical averaging
  - multiple watermarking
  - spoofing
  - …

Attacks (4)

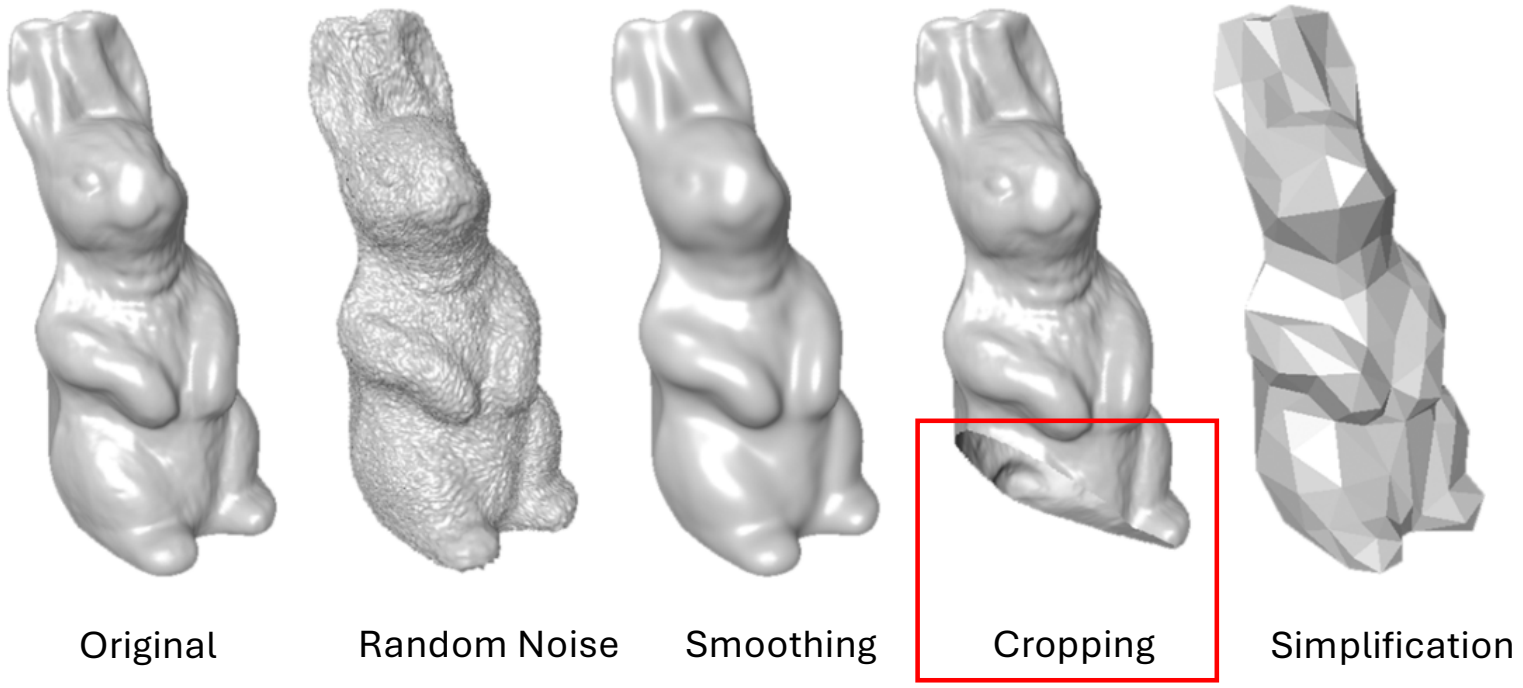
- Example 1 (cropping attack against a digital image):

Original

Cropped

Attacks (4)

- Example 2 (various attacks against a 3D mesh):



compare this with the case of digital images

Source: K. Wang, G. Lavoue, F. Denis, A. Baskurt, “A Comprehensive Survey on Three-Dimensional Mesh Watermarking”, IEEE Transactions on Multimedia, Vol. 10, No. 8, pp. 1513-1527, Dec. 2008

Security Implications

Security Implications (1)

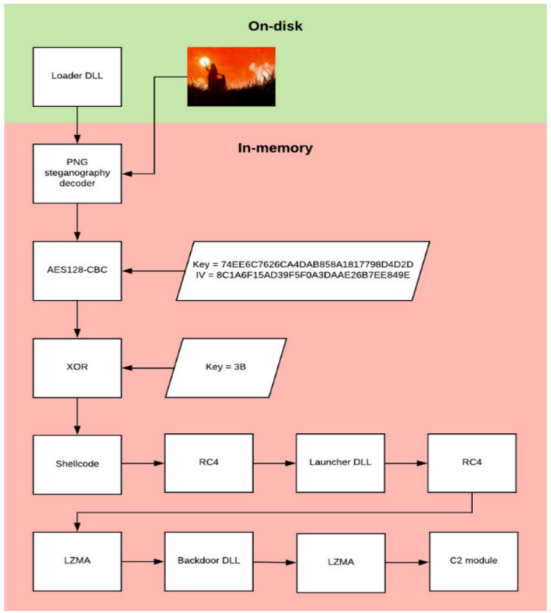
- Watermarking is **tightly-coupled** with:
  - information hiding
  - steganography
- If you can **hide data** for IP, fingerprinting, and integrity **you can**:
  - track (unaware) individuals
  - hide malicious contents
  - smuggle secrets
  - industrial espionage
  - …
- In essence:
  - **opportunity makes the thief!**

Security Implications (2)

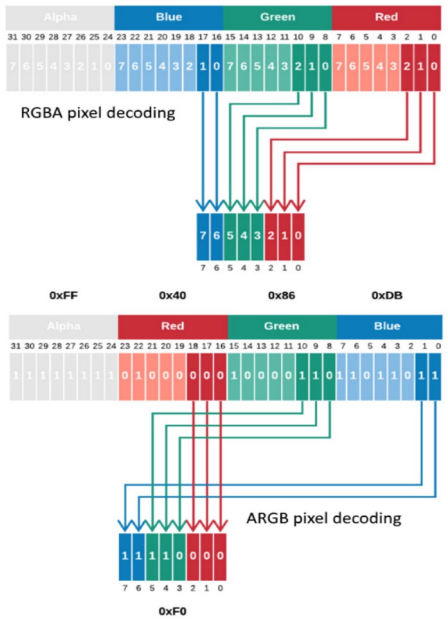
- Despite the **effort** of many **security experts** and **researchers**:
  - **countermeasures** are progressively showing **limitations**
  - only a **fraction** of threats is **detected**
  - malware increasingly operates **undisturbed** for **longer timeframes**
- How can malware developers **avoid detection** for long periods?
- Giving a final **answer** is very **hard**, but:
  - steganography and information hiding
  - covert channels
- Attackers usually borrows good ideas from other fields:
  - watermarking techniques should be considered double-edged

Security Implications (3)

- Key idea under **stegomalware** (*steganographic malware*):



Payload decoding process



OceanLotus(*) uses steganography

Source for OceanLotus: https://www.bleepingcomputer.com/news/security/oceanlotus-apt-uses-steganography-to-load-backdoors/

(*) L. Caviglione, W. Mazurczyk, “Never Mind the Malware, Here’s the Stegomalware”, IEEE Security & Privacy, vol. 20, no. 5, pp. 101-106, Sept.-Oct. 2022

Security Implications (4)

- Example of OceanLotus (*encrypted payload hidden in a .png*):

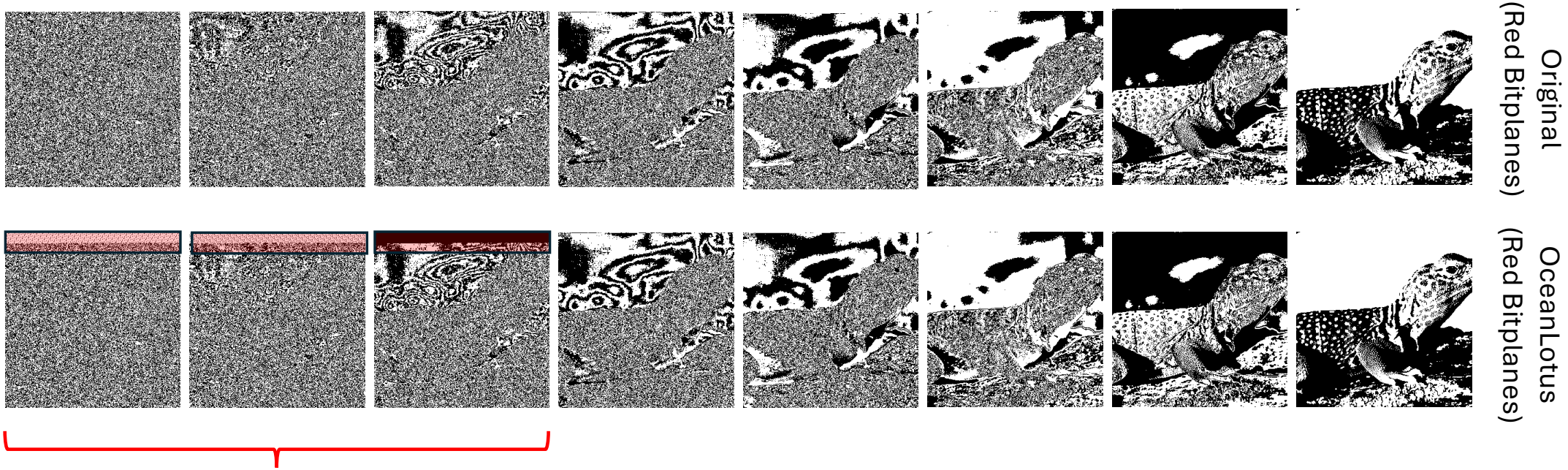
Original Image

OceanLotus Image

Examples/Digital Image/LSB Ocean Lotus

Security Implications (4)

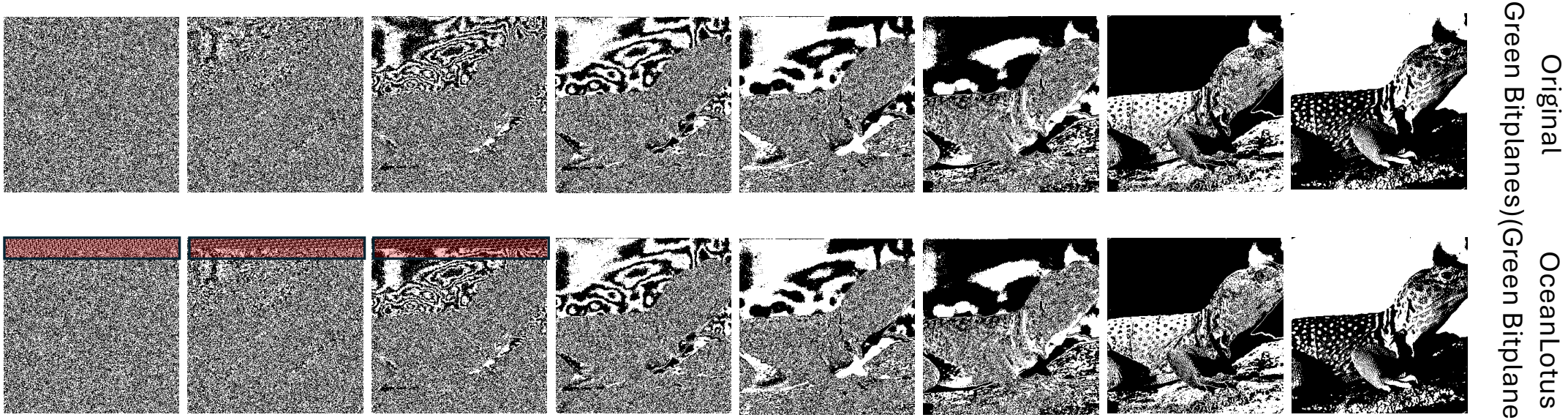
- Example of OceanLotus (*encrypted payload hidden in a .png*):



Examples/Digital Image/LSB Ocean Lotus

Security Implications (4)

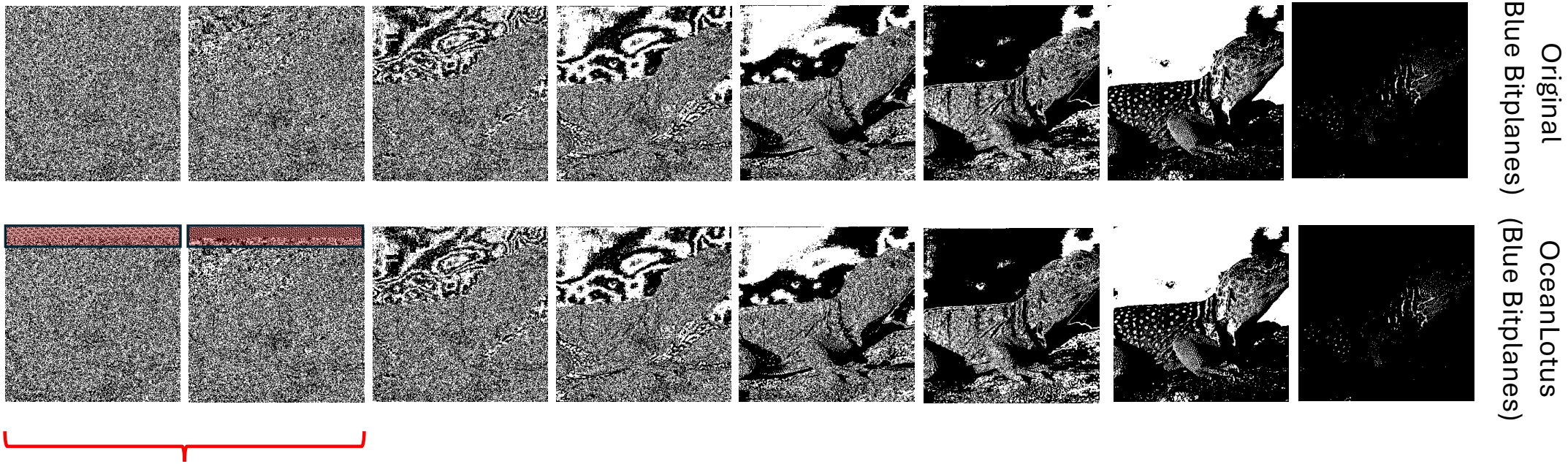
- Example of OceanLotus (*encrypted payload hidden in a .png*):



Examples/Digital Image/LSB Ocean Lotus

Security Implications (4)

- Example of OceanLotus (*encrypted payload hidden in a .png*):

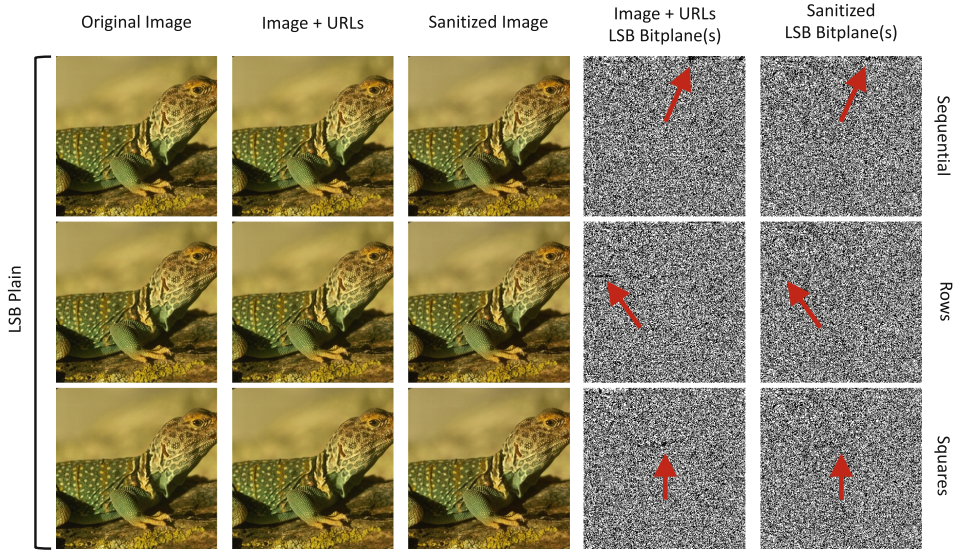


Examples/Digital Image/LSB Ocean Lotus

Open Research Challenges

Open Research Challenges (1)

- **Prevent** that a method could be used to **conceal malicious data**:
  - try to hide the minimum amount of information needed
- **Discriminate** between “good” and “bad” content:
  - already “pretty good” at detecting and sanitizing stegomalware with AI(*)



Note: “sanitizing” an image containing some steganographic data is very close to attacking a watermark!

(*) A. Liguori, M. Zuppelli, D. Gallo, M. Guarascio, L. Caviglione, “Erasing the Shadow: Sanitization of Images with Malicious Payloads using Deep Autoencoders”, 27th International Symposium on Methodologies for Intelligent Systems, pp- 115 – 125, Poitiers, France, June 2024

Open Research Challenges (1)

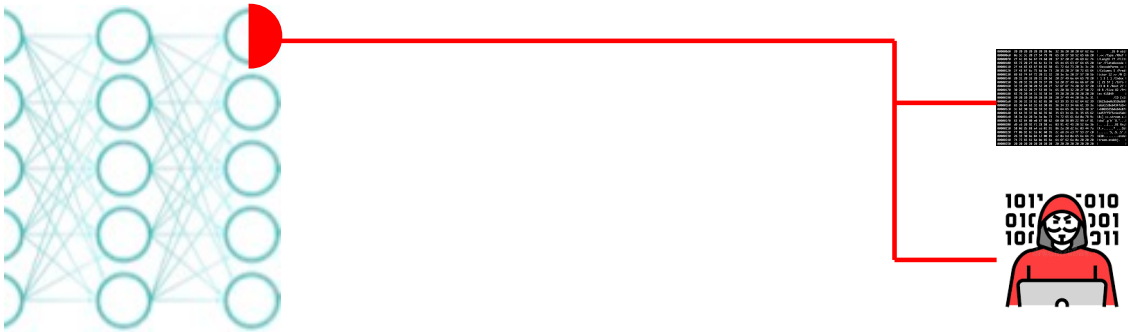
- **Prevent** that a method could be used to **conceal malicious data**:
  - try to hide the minimum amount of information needed
- **Discriminate** between “good” and “bad” content:
  - already “pretty good” at detecting and sanitizing stegomalware with AI(*)
- This is (again!) a **double-edge** sword:
  - good **sanitizers** could be also effective in **removing watermarks**
  - valuable for **attackers**!
- Create “**selective**” sanitizers:
  - malware is removed, watermark is preserved

Open Research Challenges (2)

- Watermarking techniques can be applied to **AI** to create a **new-wave** of stegomalware

Open Research Challenges (2)

- Watermarking techniques can be applied to **AI** to create a **new-wave** of stegomalware
- A malicious **payload** can be **cloaked** via(*):
  - **classical LSB** approaches, i.e., the last bit(s) of some model parameters are altered to encode arbitrary data



(*) T. Liu, et al., "Stegonet: Turn Deep Neural Network into a Stegomalware", Proc. of the Annual Computer Security Applications Conference, New York, NY, USA, 2020, p. 928–938

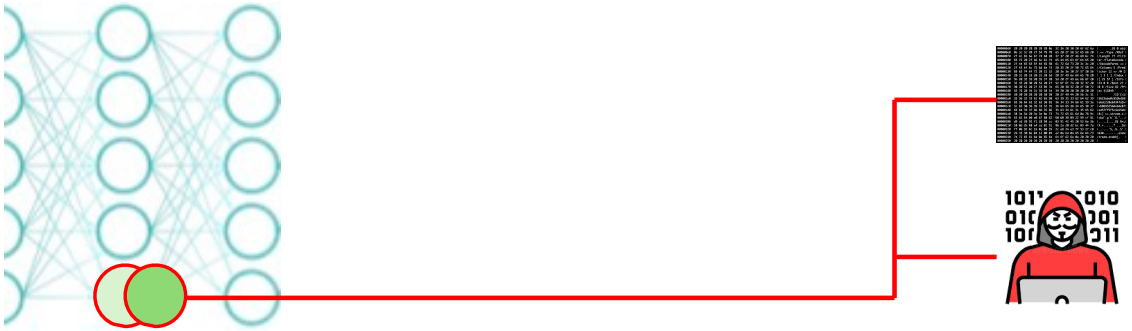
Open Research Challenges (2)

- Watermarking techniques can be applied to **AI** to create a **new-wave** of stegomalware
- A malicious **payload** can be **cloaked** via(*):
  - **exploiting resilience** of deep neural networks to introduce internal errors by overwriting the model with secret data and do not re-train the resulting “broken neurons”

(*) T. Liu, et al., "Stegonet: Turn Deep Neural Network into a Stegomalware”, Proc. of the Annual Computer Security Applications Conference, New York, NY, USA, 2020, p. 928–938

Open Research Challenges (2)

- Watermarking techniques can be applied to **AI** to create a **new-wave** of stegomalware
- A malicious **payload** can be **cloaked** via(*):
  - **mapping** techniques, i.e., arbitrary information is placed by altering parameters with similar or closest values



(*) T. Liu, et al., "Stegonet: Turn Deep Neural Network into a Stegomalware", Proc. of the Annual Computer Security Applications Conference, New York, NY, USA, 2020, p. 928–938

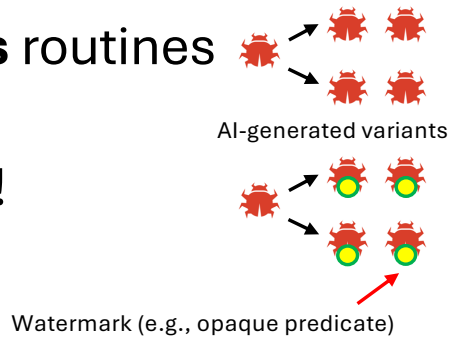
Open Research Challenges (3)

- Watermarking techniques are in general **very specialized**, but:
  - based on **recurring** operations (e.g., moving, reordering, and delaying)
  - sometimes with a good level of **abstraction** (e.g., LSB)
- Finding some **embedding patterns**(*) is desirable to:
  - pursue optimization
  - anticipate attacks or misuses observed in similar cloaking algorithms
  - create general templates
  - fix the terminology and unify the literature
- A major optimization concerns the **green-AI**:
  - watermarking large-scale datasets could be computing/energy consuming

(*) S. Wendzel, L. Caviglione, W. Mazurczyk, A. Mileva, J. Dittmann, C. Krätzer, K. Lamshöft, C. Vielhauer, L. Hartmann, J. Keller, T. Neubert, “A Revised Taxonomy of Steganography Embedding Patterns”, in Proc. of the 16th International Conference on Availability, Reliability and Security, Vienna, Austria, pp. 67:1 – 67:12, Aug. 2021

Open Research Challenges (4)

- Watermarking is useful to give an **advantage** to defendants
- Part of the **ongoing research** tries to hide information to identify:
  - AI-generated contents
  - fake news and deep fakes
  - large-scale ingestion of copyrighted data
- For the specific case of LLMs:
  - very efficient in generating code
  - used to create multiple (unbounded) mutations of **malicious** routines
  - this can challenge signature-based detection mechanisms
  - watermarking LLMs-generated code could restore a balance!



Open Research Challenges (5)

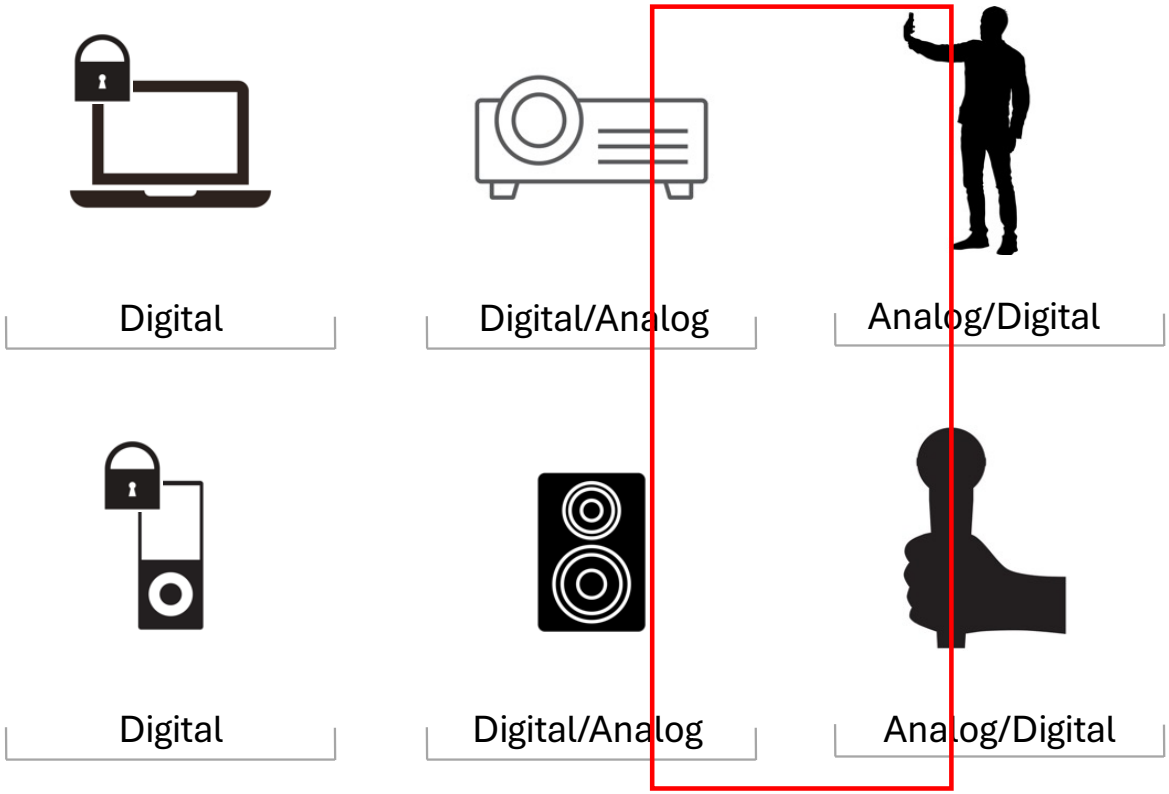
- Recommendations and regulations(*):
  - mostly on a **country-by-country** basis
  - nowadays a vivid area of **discussion** and research
- Some examples:
  - **EU** Artificial Intelligence Act: recommendations on watermarking contents
  - **US** - Ensuring Safe, Secure and Trustworthy AI: guidelines for watermarking audio and visual contents
  - **China**: many AI services should display proper information to make users aware of the content creator (e.g., human vs machine)
  - **Microsoft**: announced watermarking capabilities for their tools, especially to fight diffusion of fake contents

EU Artificial Intelligence Act - https://artificialintelligenceact.eu

(*) J. Hwang, S. Oh, “A Brief Survey of Watermarks in Generative AI”, Proc. of the 14th International Conference on Information and Communication Technology Convergence, Jeju Island, Korea, pp. 1157-1160, Oct. 2023

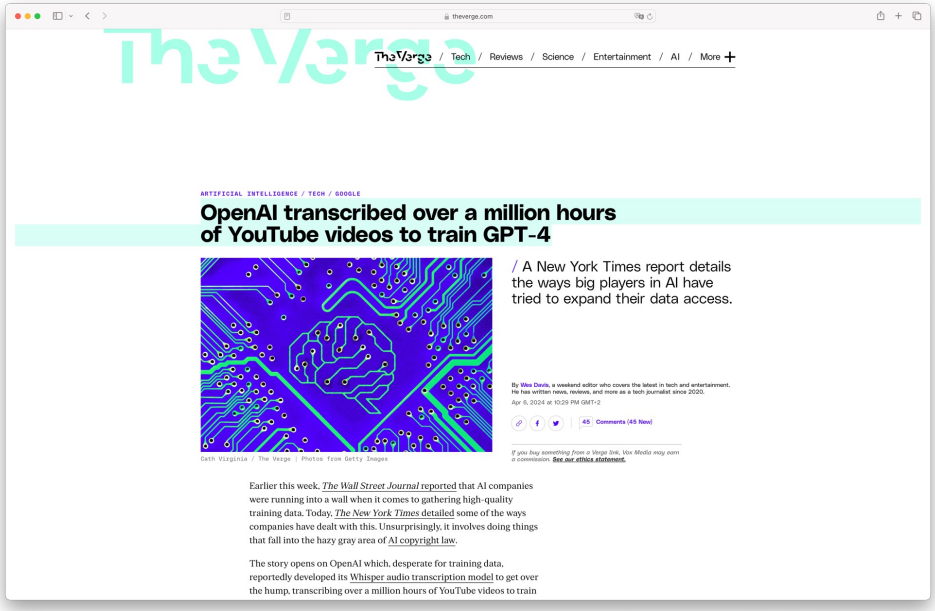
Open Research Challenges (6)

- Digital watermarking is susceptible to **analog holes**

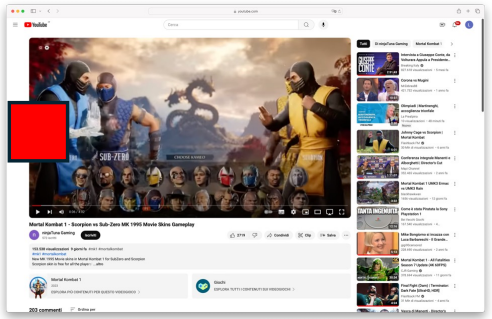


Open Research Challenges (6)

- **Cross-domain** watermark for facing new holes?



Watermark



Lore sum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo cons t. Duis aute irure dolor in reprehenderit in voluptate velit e cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium premque laudantium, totam rem aperiam, eaque ipsa quae ab illo ventore veritatis et quasi architecto beatae vitae

OpenAI transcribed over a million hours of YouTube videos to train GPT-4: https://www.theverge.com/2024/4/6/24122915/openai-youtube-transcripts-gpt-4-training-data-google

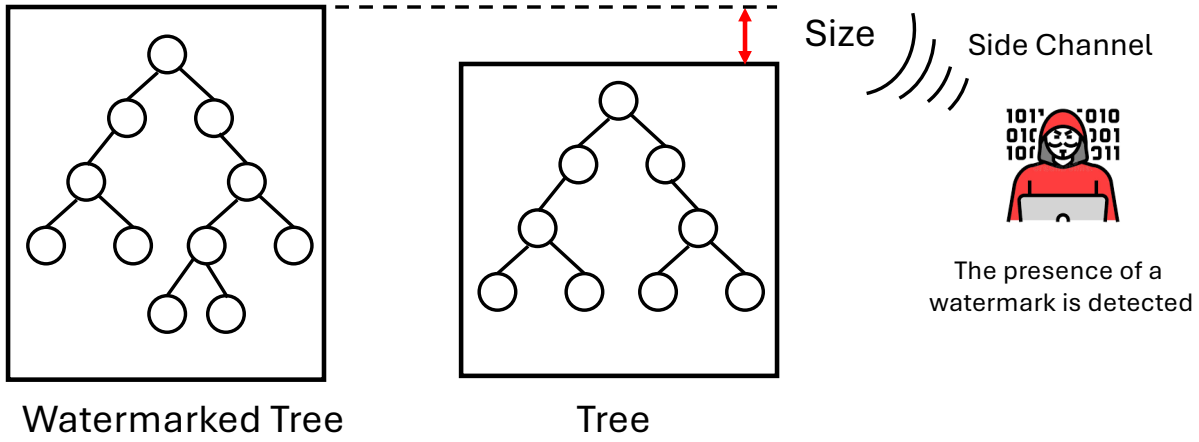
Takeaways

Takeaways (1)

- Watermarking is tightly-coupled with:
  - **the carrier**: there is no a unique hiding strategy
  - **the application**: how data will be used/retained plays a major role
  - **the processing pipeline**: it may disrupt the information
  - **the attack model**: each threat requires a specific solution

Takeaways (2)

- Double-edged cases everywhere:
  - the need of being sometimes **robust**, sometimes **fragile**
  - offering solutions to **unwanted** tracking – profiling
  - ability of an attacker of “piggyback” a method to **conceal information**
  - **side-channels** and **data-leakages**



Takeaways (3)

- Watermarking is a very **heterogeneous** area borrowing ideas from:
  - steganography and steganalysis
  - information hiding and software/cyber security
  - overlapping and “incompatible” fields

Conclusions

Conclusions (1)

- Watermarking can be used for:
  - defense, e.g., to track or protect software assets and AI models
  - attack, e.g., to hide malicious data or create covert communication paths
- No **one-size-fits-all** mechanisms for:
  - the embedding process
  - attacks
  - applications
  - design of countermeasures against removal (not covered in this class)
- Data can be hidden (almost) in any media or digital object

Conclusions (2)

- **Information hiding** is a vivid research area expected to **grow**:
  - driven by **AI** (and satellites areas)
  - to tame the **impact** of data and technology over **people**
- Challenges in watermarking are not limited to:
  - cybersecurity
  - regulations
  - performances
- **We only scratched the surface!**
- Thank you for listening
- Questions?

https://github.com/lucacav